Noname manuscript No.
(will be inserted by the editor)

Auto-inpainting Heritage Scenes: A Complete Framework for Detecting and Infilling Cracks in Images and Videos with Quantitative Assessment

Milind G. Padalkar · Manjunath V. Joshi

Received: date / Accepted: date

Abstract The need for preservation of cultural heritage has desiderated research on digitally repairing the photographs of damaged monuments. In this paper, we first propose a technique for automatically detecting the cracked regions in photographs of monuments. Unlike the usual practice of manually selecting the mask for inpainting, the detected regions are supplied to an inpainting algorithm. Thus, the process of digitally repairing the cracked regions that physical objects have, using inpainting, is completely automated. The detection of cracked regions is based on comparison of patches, for which we use a measure derived from the edit distance, which is a popular string metric used in the area of text mining. Further, we extend this method to perform inpainting of video frames by making use of the scale-invariant feature transform and homography. We consider the camera to move while capturing video of the heritage site, as such videos are typically captured by novices, hobbyists and tourists. Finally, we also propose a video quality measure to quantify the temporal consistency of the inpainted video. Experiments have been carried out on videos captured from the heritage site at Hampi, India.

This work is a part of the project *Indian Digital Heritage (IDH) - Hampi* sponsored by Department of Science and Technology (DST), Govt. of India (Grant No: NRDMS/11/1586/2009/Phase-II).

Milind G. Padalkar
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat, India
E-mail: milind_padalkar@daiict.ac.in

Manjunath V. Joshi
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat, India
Tel.: +91-79-30510611
E-mail: mv_joshi@daiict.ac.in

Keywords Auto-inpainting · Heritage scenes · Crack-detection · Edit distance · SIFT · Homography.

1 Introduction

Historic monuments and heritage sites across the world are important sources of knowledge, depicting the evolution of mankind. These are not only irreplaceable assets that signify the culture and civilization of the past, but also masterpieces of accomplishments that symbolize the human potential. It is for this reason that globally many organizations have taken up the initiative to safeguard and preserve the heritage sites. Over the centuries, the heritage sites have witnessed a number of natural calamities and sabotage, resulting in their present ruined condition. Access to many such heritage sites is restricted, fearing the risk of further damage by visitors. One may think of physically renovating the heritage sites to preserve them. However, the renovation may not only pose danger to the undamaged monuments, but may also fail to mimic the skilful work of history. It would be interesting to have a heritage site reconstructed digitally, as such a process avoids physical contact to the monuments. The digitally reconstructed heritage site may then provide an unrestricted access for viewing the monuments in their entirety. Also, in today's world, preservation of the digitally reconstructed monuments would be inexpensive.

Digital reconstruction requires repairing of the damaged regions in a plausible manner. This task can be achieved using various inpainting techniques [4], [9], [10], [33]. Given an image and a region of interest in it, the task of an inpainting process is to fill up the pixels in this region, in such a way that either the original content is restored or the region is visually plausible in

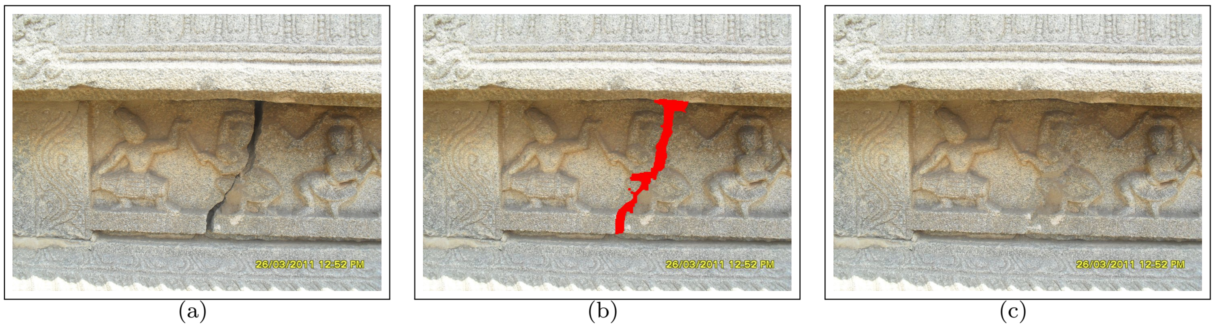


Fig. 1 Auto-inpainting cracked regions. (a) Original image of a heritage scene. (b) Automatically detected cracked region using the proposed method is shown in red color. (c) Image obtained after inpainting the detected region.

the context of the image. Digital restoration of the damaged regions in given images thus consists of two steps viz. (a) selection/detection of the regions to be modified and (b) applying a suitable inpainting algorithm on these regions.

The process of selecting the regions to be inpainted is usually subjective. One user may want some region of the image to be modified, while another user may want to modify another region in the same image. Hence, for an inpainting algorithm, the regions to be inpainted are usually selected manually. However, when looking at heritage monuments, there is a consensus about the desire to view these in their undamaged form. In particular, the damage involving cracked regions diminishes the attractiveness of the monuments and one would crave these to be seamlessly eliminated. One such example is illustrated in figure 1.

The result of an inpainting algorithm is sensitive to the selection of the region to be modified. The exact selected area may vary for different users if the selection of cracked regions in photographs of monuments is done manually. Therefore, apart from being subjective, the process of selecting the cracked regions is also an enervating task. This necessitates an exploration for a technique that can automatically detect the cracked regions in images of monuments, which is a critical and challenging problem for the success of digital reconstruction of heritage sites [11], [12], [19]. An automatic detection of cracked regions also proves useful in reconstruction and repair of digitized 3D models. One can use these digitized 3D models for creating walk-through applications [3], [20], [40]. Furthermore, automatic detection of cracked regions will facilitate inpainting to be performed on-the-fly for creating efficient immersive navigation/digital walk-through systems.

This paper contributes by proposing a novel technique to auto-inpaint photographs of damaged historic monuments and its extension to inpaint videos. Note that the application is to actually restore a heritage scene, i.e., digitally repair cracks that physical objects have. Thus we are not talking about image restoration, but about object completion. In other words, unlike the techniques that detect an external damage or defect due to alteration of a photograph, the proposed method aims to detect and inpaint the cracked regions in the photographed scenes/objects. The cracks could be developed over a period of time due to environmental effects or due to manual destruction. The paper also proposes a video quality metric to measure the temporal consistency of the inpainted video.

The detection of cracked regions uses similarity of non-overlapping adjacent patches as a cue. In videos, the cracked regions detected in a frame are inpainted and then tracked across subsequent frames for maintaining the visual continuity. Videos of heritage scenes captured with a moving camera by novices, hobbyists and tourists usually contain rigid objects. In such a situation, to track the detected cracked regions in subsequent frames, we use homography [17] estimated by matching scale invariant feature transform (SIFT) keypoints [23].

The rest of the paper is organized as follows: Literature review is presented in section 2. Our proposed technique for detecting cracked regions is discussed in detail in section 3 and its extension to inpaint videos is described in section 4. The proposed temporal consistency measure is given in section 5. The experimental setup and the reported results are given in section 6, followed by conclusion in section 7.

2 Literature Review

Over the past two decades, image inpainting has been an active area of research. Techniques based on connecting level lines [4], [16], [26], [28], [45], [47], exemplar based methods [9], [10], methods based on image gradients [33], probabilistic structure estimation [39], methods using depth and focus [27], etc. have been influential. However, these methods are semi-automatic, i.e. the regions to be inpainted are required to be manually selected by the users.

The literature reports only a few inpainting techniques that also facilitate the automatic detection of the regions to be inpainted [1], [7], [42]. Chang et al. [7] proposed a method to detect damage in images due to colour ink spray and scratch drawing. Their method is based on the use of several filters and structural information of damages. Tamaki et al. [42] address the detection of visually less important string-like objects that block user's view of a discernible scene. Their method, however, is restricted to the detection of only those occluding objects that are long and narrow and highly contrasted in intensity with respect to the background. Amano [1] presents a correlation-based method for detecting defects in images. This method relies on correlation between adjacent patches for detection of defects i.e. small number of regions disobeying an “image description rule”, complied by most local regions. The method works well for detecting computer-generated superimposed characters having uniform pattern.

All the above mentioned techniques are suitable for detecting actual damage or alteration caused to a photograph. These techniques do not address the identification of damage to the objects or scenes that are photographed. In this direction Parmar et al. [31] proposed a technique which uses matching of edge-based features with pre-existing templates to distinguish vandalized and non-vandalized regions in frontal face images of monuments at heritage sites. However, their inpainting results are highly dependent on the selected templates and their method is restricted to frontal face images of monuments. The template creation of both vandalized and non-vandalized regions may not be practically realizable for such images and therefore the detection process may lead to undesired results. Another method to identify and inpaint the defaced regions in statues is proposed in [29]. The method matches templates by comparing the texton features and is again limited to frontal faces. Turakhia et al. [43] proposed a method to automatically inpaint cracks in images of heritage monuments. Their method relies on edge detection and tensor voting to detect cracks.

The technique proposed in [30] compares overlapping adjacent patches for similarity. The patches are compared in the singular value decomposition (SVD) domain and use an image-dependent threshold to identify cracked regions. However, overlapping patches make redundant comparisons due to which implementation slows down. Recently, Cornelis et al. [8] have proposed a method for virtual restoration of paintings. The method is flexible as the user may set parameters to suit the input. However, it is suitable only for the detection of fine cracks that appear on paintings.

Techniques for micro-crack detection in concrete can be found in [2], [34], but one may note that these require special imaging conditions. The actual surface is polished, impregnated with a special dye and then photographed using microscope. In this way defects of material including micro-cracks, transition zones, porous areas and air-bubbles are highlighted, generating a high contrast image. A method for crack detection in pavement images using tensor voting can be found in the work by Zou et al. [48]. The performance of their technique is heavily dependent on the accuracy of generation of crack-pixel binary map that acts as an input to the tensor voting framework.

For inpainting in videos, a method has been proposed by Patwardhan et al. [32]. Their technique considers a static background with a moving foreground, any of which could fall under the region to be inpainted. First, the occluded foreground patches are filled up using motion-inpainting. The background patches which are visible in other frames are then directly copied. Finally, any missing region is filled up using the exemplar-based inpainting approach [10]. It may be noted that, in this approach the users need to manually specify the objects or regions that are to be inpainted. Also, many constraints are placed on the camera motion.

3 Proposed approach for detection of cracked regions

Visual discontinuities like damaged regions in a photographed scene/object attract attention of the human visual system. The cracked areas are the breaks splitting the objects which were developed over a period of time due to natural calamities or manual destruction. Inpainting these shall improve the visual appearance and enable one to view the photographed scene/object in an undamaged form. Here, we propose a novel technique for automatically detecting such cracked regions.

Cracks are typically characterised by dark areas in an image. These can be easily identified by human beings but pose difficulty to computers. In trivial cases, simple thresholding is sufficient for detecting the cracks. However, in general, the subtle variation in pixel intensities makes it challenging to detect the cracked regions. In what follows, we describe a method for crack detection by enhancing the dark regions and comparing non-overlapping patches. The patches are compared using a distance measure inspired from the edit distance [44] which is successfully used in the area of text mining for comparing strings. The distance measure is such that

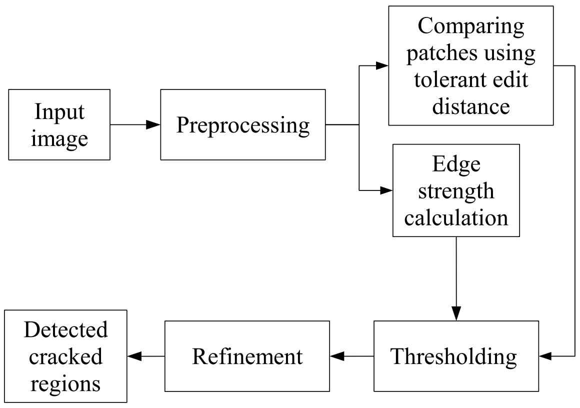

Input image
Preprocessing
Comparing patches using tolerant edit distance
Edge strength calculation
Thresholding
Refinement
Detected cracked regions

Fig. 2 Proposed approach for detecting the cracked regions.

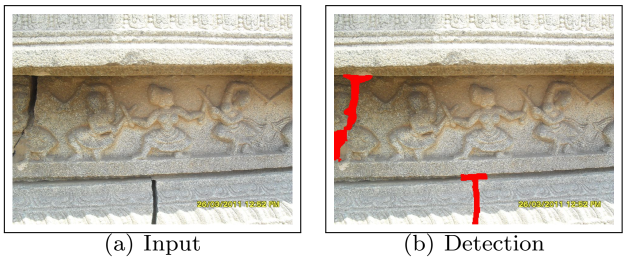


Fig. 3 Proposed detection of cracked regions. The detected regions are shown in red color.

it avoids penalizing trifle differences between the corresponding pixels of the compared patches. The patch penalty along with average edge strength within the patches is used to detect the cracked regions. The proposed method is shown in figure 2 and the steps involved are described as follows.

3.1 Preprocessing

Consider a normalized intensity image I_0 of size $M \times N$. Since the cracked regions are dark, low-intensity pixels are more likely to be part of a crack. A weight matrix I_w is constructed such that dark pixels have higher weights, given by,

$$I_w(x, y) = \exp(-I_0(x, y)), \tag{1}$$

where (x, y) denote the pixel coordinates. The weights in I_w are multiplied to the corresponding pixels in I_0 and the resulting image is eroded to obtain I_v. The erosion operation is performed so that the narrow dark regions grow sizeably for proper detection, which may otherwise remain undetected during further processing. The results of preprocessing on the input image shown in figure 3(a) are depicted in figure 4.

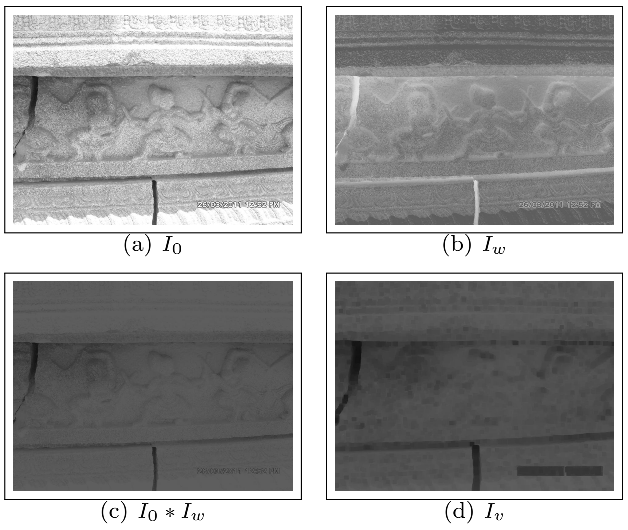


Fig. 4 Preprocessing of the input image shown in figure 3(a).

3.2 Comparing patches using tolerant edit distance

Since the cracked regions exhibit noticeable dissimilarity with respect to the neighbouring regions, we intend to mark them out by comparing adjacent non-overlapping patches in the image I_v. A simple method for comparison is to calculate sum of absolute difference or sum of squared difference (SSD) across corresponding pixels of the compared patches. These measures are, however, sensitive to noise and may give a high error even for visually similar patches, which is evident in figure 5. Moreover, comparing a patch with its spatially shifted version also gives high error, where in fact both are visually identical. Thus, it becomes difficult to separate the cracks from the surroundings using a threshold.

In string matching, shifting errors are overcome using the edit distance [44]. Edit distance is a string metric that gives the count of operations required for transforming one string into another. The transformation is achieved by comparing the characters of first string with that of the second string and performing an appropriate operation. Here, the valid operations on comparing a pair of characters are insertion, deletion and substitution. For example, consider two strings "books" and "loops". Here only two operations, both substitutions viz. "b" to "l" and "k" to "p" are required for the transformation. Hence the edit distance between "books" and "loops" is 2. Likewise, for transforming "books" to "oops" we again require two operations, a deletion and a substitution, giving an edit distance equal to 2.

Now, in order to compare patches, consider the lexicographical ordering of two patches synonymous to two strings and each pixels synonymous to characters of the respective strings. If we calculate the edit distance, it

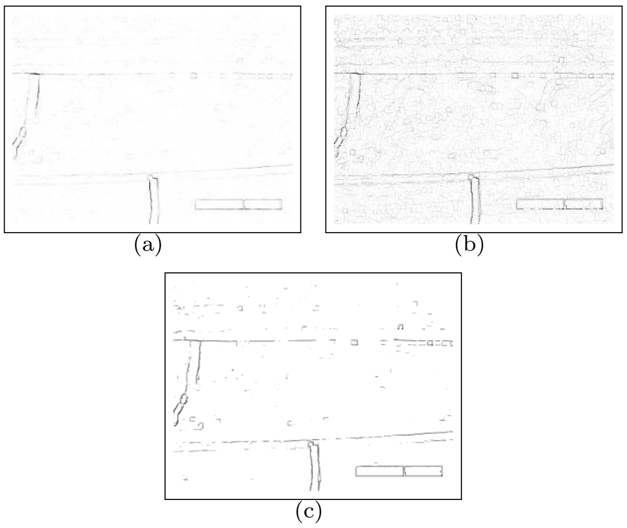

(a) (b) (c)

Fig. 5 Comparison of (a) sum of absolute difference image, (b) sum of squared difference image and (c) tolerant edit distance image I_{tED} for tolerance $\delta_t = 10$. Patch size is 3×3. With the input image of size 684×912 we have I_{tED} of size 227×303. Here, an enlarged, intensity inverted version is shown for clarity.

Algorithm 1 Calculation of tED

```
% For vectors $v_1$ and $v_2$ with lengths $|v_1|$ and $|v_2|$, respectively and $\delta_t$ as tolerance value,

% Initialization
$D[0,0] := 0$
for $i := 1$ to $|v_1|$ do $D[i,0] := i$ end for
for $j := 1$ to $|v_2|$ do $D[0,j] := j$ end for

% Required operation: substition, insertion or deletion
for $i := 1$ to $|v_1|$ do
    for $j := 1$ to $|v_2|$ do
        $m_1 := D[i-1, j-1] + C(v_1[i], v_2[j], \delta_t)$
        $m_2 := D[i-1, j] + 1$
        $m_3 := D[i, j-1] + 1$
        $D[i,j] = \min(m_1, m_2, m_3)$
    end for
end for

% Result
return tED := $D[|v_1|, |v_2|]$

% Comparison function: $C(v_1[i], v_2[j], \delta_t)$
if $|v_1[i] - v_2[j]| \leq \delta_t$ then
    $C(v_1[i], v_2[j], \delta_t) := 0$
else
    $C(v_1[i], v_2[j], \delta_t) := 1$
end if
```

would give the number of operations required to transform one patch to another. A smaller value of edit distance conveys less number of operations and in turn higher similarity of the patches. However, in the presence of noise, the edit distance will still be higher. This is because the substitution operation penalizes the mismatch of compared characters.

In order to overcome the noise sensitivity of edit distance, a tolerance can be used for the substitution operation. In other words, if the difference between the compared characters falls within some tolerance value, the characters can be considered as equivalent and, therefore, no penalty is given by the substitution operation. We call the edit distance with such a substitution operation as tolerant edit distance (tED). The tED thus gives a measure of similarity between patches, in the presence of noise and spatial shift.

Consider patches of size $m \times n$. Then, patch Φ_p at pixel p with coordinates (x, y) in the image I_v, consists of pixels with coordinates (X, Y), such that $X = x, \dots, x+m-1$ and $Y = y, \dots, y+n-1$. For patch Φ_p, the right and bottom non-overlapping adjacent patches are Φ_r and Φ_s at pixels $r = (x, y+n)$ and $s = (x+m, y)$, respectively. Let the pixels of patches Φ_p, Φ_r and Φ_s be rearranged using lexicographical ordering to form vectors v_p, v_r and v_s, respectively. We then calculate the tED between the pairs v_p, v_r and v_p, v_s, the average of which is assigned to patch Φ_p. The tED is calculated using the edit distance calculation method described in [44], along with a tolerance value[1] δ_t, as given in algorithm 1. The tED is calculated for all the patches for which there exist both left and bottom non-overlapping adjacent patches. The calculated tED values are used to form an image I_{tED}. The image I_{tED} when multiplied with an edge strength image makes it easier to detect the cracked regions. Figure 5(c) shows the image I_{tED} corresponding to image I_v depicted in figure 4(d).

3.3 Edge strength calculation

Since the cracked regions are distinct from their neighbouring regions, these exhibit higher edge strengths. In order to give preference to patches having higher edge strength, we now calculate the normalized gradient magnitude of every pixel in the image I_v. Let I_g represent the image consisting of normalized gradient magnitudes. The gradient magnitude along the boundary of the cracked regions may vary and therefore the pixels of a cracked region may not have a unique edge strength. In order to assign a unique edge strength to each cracked region, we intend to identify the regions disconnected by weak gradient magnitudes.

The boundary of the cracked region within a small (say 3×3) patch would be similar to a horizontal, vertical, diagonal or anti-diagonal line. Since the gradient

[1] The Details of selecting a suitable tolerance value δ_t are given in section 6.

0	0	0
1	1	1
0	0	0

(a)

1	0	0
0	1	0
0	0	1

(b)

0	1	0
0	1	0
0	1	0

(c)

0	0	1
0	1	0
1	0	0

(d)

Fig. 6 Line filters. (a) Horizontal, (b) main diagonal, (c) vertical and (d) anti-diagonal.

of the boundary could vary, within this small patch the pixels which are part of a horizontal, vertical, diagonal or anti-diagonal line can be detected using the corresponding line filters. To achieve this, we use a set of four 3×3 line filters shown in figure 6. By convolving the image I_g with these filters, the maximum response at each pixel is recorded to create an image I_m.

In all our experiments we observed that the pixels around the boundary of the cracked regions have a low non-zero response to the line filter. Because of this, the disjoint cracked regions get connected while performing unique edge strength assignment. To avoid such a situation, the filter responses having lower values are required to be discarded using an image dependent threshold. Since the response to line filters is not expected to vary significantly for pixels in the cracked regions, a threshold with respect to the maximum response can be used. Setting the threshold to 0.1 times the maximum response was found appropriate for discarding the low non-zero responses which were responsible for connecting the disjoint cracked regions. The image I_m is thus refined by discarding the low responses as follows.

$$I_m(x,y) = \begin{cases} 0, & \text{if } I_m(x,y) < 0.1 * \max(I_m), \\ I_m(x,y), & \text{otherwise.} \end{cases} \quad (2)$$

The image I_m is morphologically closed using a 3×3 structuring element and the connected components are detected. The gradient magnitude image I_g is now updated such that the highest gradient magnitude within each connected component is assigned to all the pixels within the respective component. Updating I_g in this manner enables us to assign a unique edge strength value to distinct components. The edge strength image I_e is now constructed by taking the normalized sum of I_g and I_w.

As mentioned earlier, the image I_{tED} when multiplied with the edge strength image I_e makes it easier to detect the cracked regions. Therefore, to every patch Φ_p for which tED is calculated, the average of edge strengths of all the pixels within the patch Φ_p, and its neighbours Φ_r and Φ_s, is calculated and assigned as the edge strength. We now multiply these edge strengths with the corresponding tED to form the weighted tED image I_{tw} shown in figure 7(e).

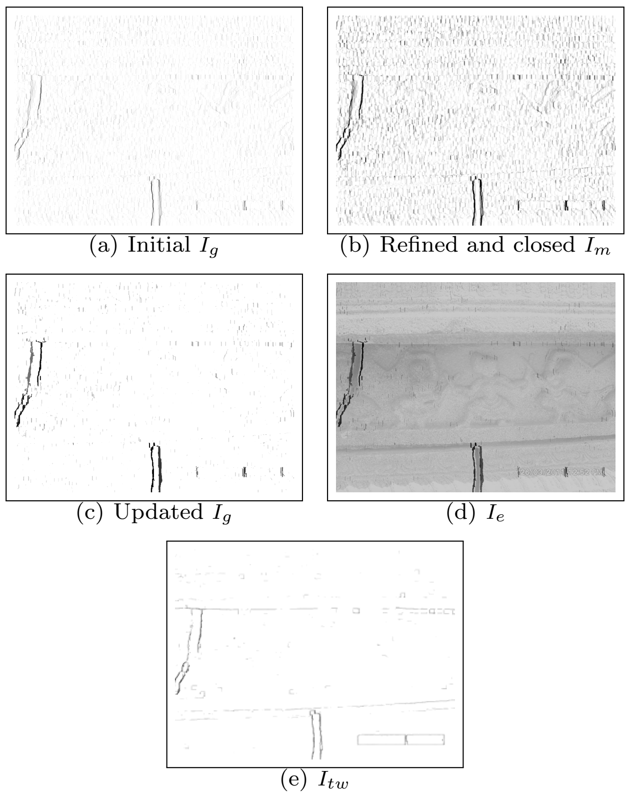
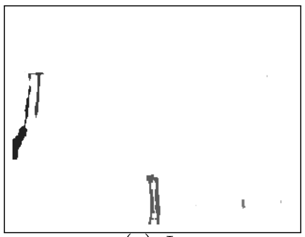


Fig. 7 Edge strength I_e and weighted tolerant edit distance images I_{tw}. Sizes of I_g, I_m and I_e are the same as that of I_0, while I_{tw} and I_{tED} are of the same size. Here, enlarged and intensity inverted version of I_{tw} is shown for clarity.

3.4 Thresholding

By multiplying the tED image I_{tED} with the edge strength image I_e, we ensure that only strong crack-boundaries are retained. Since the tED image I_{tED} has higher values at the boundary of the cracked regions, the same holds true for the weighted tED image I_{tw}. In order to fill the gap between the boundaries, a morphological closing operation is applied on I_{tw}, with the size of the structuring element depending on the size of I_v. The morphologically closed image I_{tw} is now multiplied with the resized version of the weight matrix I_w to obtain an intermediate image I_{wc}.

In order to assign unique values to different objects for segmentation in image I_{wc}, we employ the method used for updating the gradient magnitude image I_g in the previous section 3.3. Thus, by convolving the intermediate image I_{wc} with the line filters shown in figure 6, thresholding the maximum response image using equation (2) and finally applying the morphological closing operation, we obtain the image I_c, in which the connected components have unique values. The image I_c obtained here is shown in figure 8(a).


(a) I_c

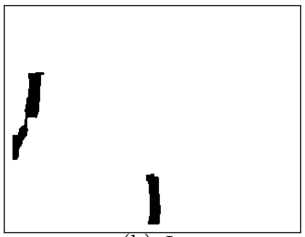

(b) I_1

Fig. 8 Initial detection. Image I_c is thresholded and mapped to I_v to obtain I_1. Size of I_c is same as that of I_{tED}, while I_1 and I_v are of the same size. Here, enlarged, intensity inverted version of I_c is shown for clarity.

Higher the value of a region in I_c, more likely it is to be a crack. Thus, the regions with values lower than a threshold T need to be discarded. Let the V denote the array consisting of k unique values in I_c arranged in ascending order. Then, inspired by the threshold selection method for matching SIFT features given in [23], we estimate the threshold T based on I_c as given below in algorithm 2.

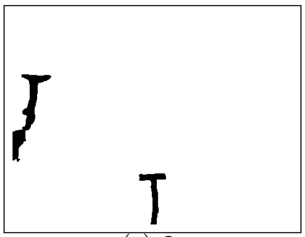
Algorithm 2 Selection of threshold T

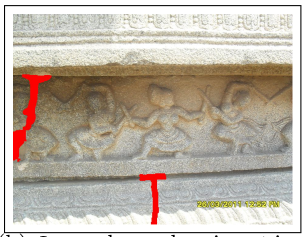
```
% Initialize
$T := V[k]$

% Update
for $i := k - 1$ to 1 do
    if $V[i] < 0.2$ then
        break
    end if
    if $(\frac{V[i]}{V[i+1]}) \geq (\frac{V[i-1]}{V[i]})$ then
        $T := V[i]$
    end if
end for

% Result
return $T$
```

Once T is calculated, the image I_c is thresholded using the following equation (3).

$$I_c(x,y) = \begin{cases} 1, \text{ if } I_c(x,y) \geq T, \\ 0, \text{ otherwise.} \end{cases} \quad (3)$$

The thresholding operation is followed by morphological closing to fill in gaps between nearby disjoint regions. Note that binary image I_c of size $(\frac{M}{m} - 1) \times (\frac{N}{n} - 1)$ is obtained by operating on $m \times n$ sized non-overelapping patches in the grayscale image I_v of size $M \times N$. Here, the patches located at (A, B), $(A, B + n)$ and $(A + m, B)$ in the image I_v are used to obtain the value at pixel (a, b) in the binary image I_c, such that $A = (a - 1) * m + 1, \ldots, a * m$ and $B = (b - 1) * n + 1, \ldots, b * n$. Since we need the output binary image to have the same size as that of the input grayscale image I_v, we need an inverse mapping from pixels in I_c to corresponding patches in I_v. For this purpose, consider an image I_1 having size same as that of I_v i.e. $M \times N$. The inverse mapping is now obtained by copying the values from location (a, b) in the binary image I_c to locations (A, B), $(A, B + n)$ and $(A + m, B)$ in I_1. This inverse mapping is performed for all pixels in I_c to corresponding patches in I_1 which is of the same size as that of I_v.


(a) I_f


(b) I_f overlapped on input image

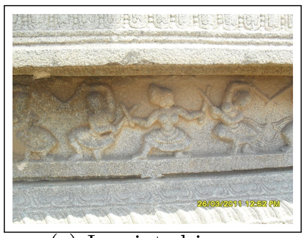

(c) Inpainted image

Fig. 9 Refined detection. (a) Final detection binary image I_f, (b) detected regions overlapped on the input image, (c) inpainted result.

A second morphological closing operation is now applied on the binary image I_1 in order to avoid splitting of the detected region after the inverse mapping. The resulting initial detection binary image I_1 is shown in figure 8(b). This gives a good estimate of the cracked regions. However, few pixels of the cracked regions which are similar to the surroundings may still remain undetected. Therefore, a refinement step is required to achieve a more accurate detection.

3.5 Refinement

The method described above relies on patch-based comparison. Therefore, the initial detection binary image I_1 localizes the cracked regions. In order to perform a more accurate detection at pixel level, sophisticated techniques are required such that a binary segmentation-based refinement around the initially detected regions can be performed. Interactive image segmentation techniques based on curve evolution, graph-cut optimization have been widely used for accurately detecting roughly marked objects. The active contour method [6] and grab-cut technique [35] require the user to manually select a region around the object of interest. By optimizing an energy function, the selection is refined to fit the object boundary.

The initially detected binary image I_1, which is detected automatically without any user interaction, can be used as input to the above mentioned interactive segmentation techniques. For refining I_1, we use the method based on active contours,[2] proposed in [6] to obtain the final detection binary image I_f. Figure 9(a) shows the final detection binary image I_f obtained on refining I_1. The detected regions overlapped on the input image are shown in figure 9(b). In order to justify the suitability of the proposed method for inpainting, we show the inpainted result in figure 9(c). For inpainting, we have used the method proposed in [10].

4 Proposed approach for auto-inpainting in videos

In order to extend the proposed crack detection method for auto-inpainting in videos, it would be intuitive to think of performing frame-by-frame detection and inpainting. This abstraction, however, in practice is a long-drawn-out process as it does not exploit the inter-frame redundancy. Moreover, in frame-by-frame processing, the pixels corresponding to cracked regions detected in one frame may not map to the pixels corresponding to the same cracked regions detected in some other frame. This is because, there may be occlusion or change in illumination across frames as the camera moves. Hence, the proposed crack detection method, which relies on properties of patches in the input frame, may not detect the exact same pixels in the two frames. This leads to large variations in the two inpainted frames for the same cracked regions, given that the inpainting task is highly sensitive to the pixels to be inpainted. Therefore, the auto-inpainted videos created by detecting and inpainting cracked regions independently in every frame, appear unstable and the effect of seam becomes visible.

Alternatively, one may think of using motion as a cue to track and inpaint the cracked regions across subsequent frames. Motion estimation and compensation have been popularly used in video compression techniques [18], [41]. Here intermediate frames are generated using independent frames and motion parameters. However, since these methods are block based, their use to inpaint videos results in blocking artefacts. Moreover, such methods are computationally expensive since the motion parameters are estimated independently for each block. A frame-to-frame transformations is, therefore, needed to track the damaged regions in subsequent frames for creating a seamlessly inpainted video.

Brown and Lowe [5] have suggested a method for automatic image stitching, wherein transformation between the images to be stitched is calculated by matching keypoints invariant to rotation, scaling and view point. Here, the transformation is considered to be projective or a homography [17]. Since the videos captured at heritage sites usually contain nearly planar rigid objects/scene with a moving camera, we can consider the video frames to be images captured from different viewpoints. Hence, the transformation between these frames can be represented by a homography.

In the proposed video inpainting method, we consider pairs of temporally adjacent frames and use the corresponding homography to track cracked regions from one frame to another. The cracked regions are detected in reference frames using the proposed method described in section 3 and then tracked to subsequent frames. Similarly, the detected cracks are inpainted in the reference frames using the technique proposed in [10] and then mapped to the tracked regions in the subsequent frames. Note that the inpainting of video frames cannot be done by simply copying objects visible in other frames, as done in [32]. This is because, an object to be inpainted in one frame also needs to be inpainted in other frames as well, which mandates the use of a hole filling technique. The proposed approach for detecting and inpainting the cracked regions in videos is shown in figure 10. The various stages involved are described below.

4.1 Homography estimation

As already mentioned, two frames of a video can be considered as images captured from different viewpoints. A frame-to-frame transformation between these frames can be estimated in the form of homography by matching the SIFT descriptors of keypoints in the two frames [17], [22]. The extraction of keypoints and corresponding SIFT descriptors[3] is performed using the method given in [23]. The SIFT features are robust to changes in contrast, illumination, rotation, scaling and view point. Let the keypoint at location (x_1, y_1) in the first frame match the keypoint at location (x_2, y_2) in the second

[2] For active contour segmentation technique, we have used the implementation available at http://www.mathworks.in/matlabcentral/fileexchange/23847-sparse-field-methods-for-active-contours

[3] An implementation for extraction and matching of SIFT keypoints and corresponding descriptor is available at http://www.cs.ubc.ca/ lowe/keypoints/

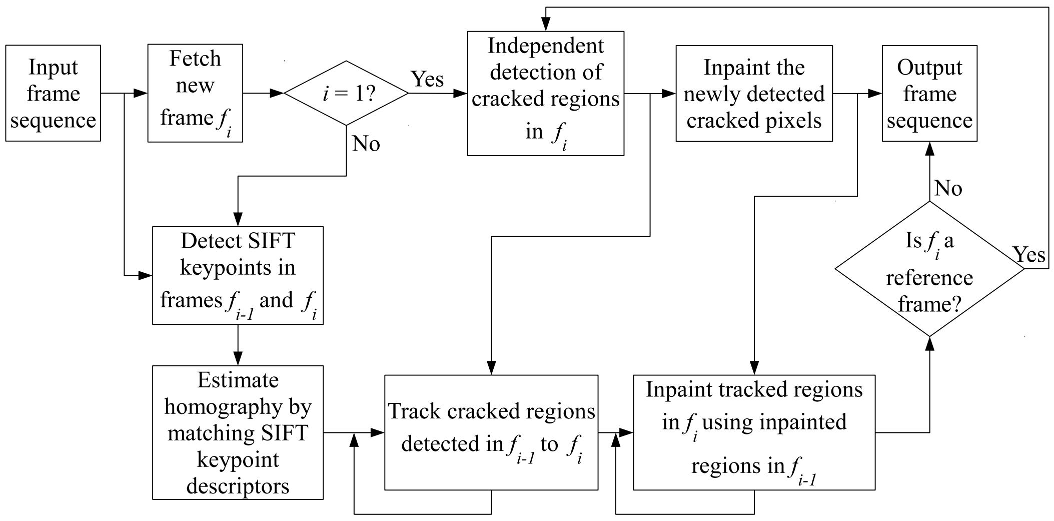

Input frame sequence
Fetch new frame f_i
$i = 1?$
Yes
No
Independent detection of cracked regions in f_i
Inpaint the newly detected cracked pixels
Output frame sequence
Detect SIFT keypoints in frames f_{i-1} and f_i
Estimate homography by matching SIFT keypoint descriptors
Track cracked regions detected in f_{i-1} to f_i
Inpaint tracked regions in f_i using inpainted regions in f_{i-1}
Is f_i a reference frame?
No
Yes

Fig. 10 Proposed approach for detecting and inpainting the cracked regions in video.

frame. For a set of such matching keypoints, the homography matrix H obeys the following relation [17],

$$\begin{bmatrix} x_2' \\ y_2' \\ z_2' \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}, \qquad (4)$$

where (x_2', y_2', z_2') are the homogeneous coordinates for the point (x_2, y_2) in the second frame such that $x_2 = \frac{x_2'}{z_2'}$ and $y_2 = \frac{y_2'}{z_2'}$, and H is a 3×3 non-singular matrix.

Using the set of matched keypoint locations, the homography matrix H is estimated using equation (4) by setting $z_2' = 1$ i.e. setting the homogeneous coordinates $(x_2', y_2', z_2') = (x_2, y_2, 1)$. Here, the random sampling consensus (RANSAC) algorithm [15] is used to iteratively eliminate the keypoint matches that do not agree with the estimate of homography matrix.[4] Figure 11 illustrates the matching of SIFT keypoints between a pair of video frames.

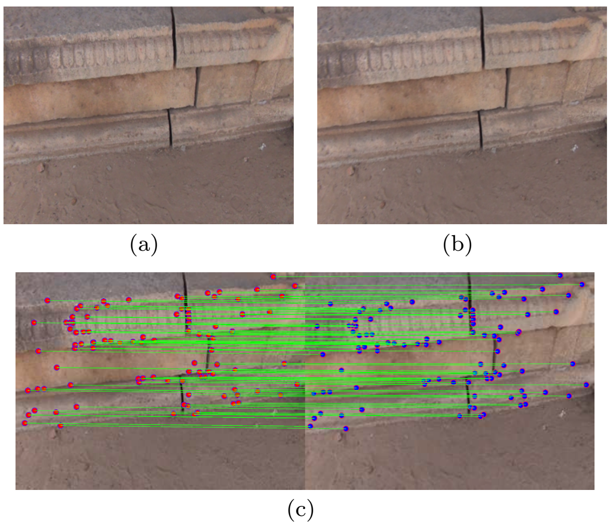


Fig. 11 Matching of SIFT keypoints. (a)–(b) Two frames of a video; (c) Pairs of matching keypoints shown by green joining lines.

4.2 Reference frame detection

A reference frame is the one in which cracked regions are detected independently. While capturing the video with a moving camera, new cracked regions may appear. If the cracked regions are detected in the first frame and tracked across all subsequent frames, the new cracks that appear as the camera moves will not be detected. Therefore, an independent crack detection needs to be performed quasi-periodically depending the camera movement. Thus, for fast camera movement, the detection needs to be performed more frequently, while for slow camera motion, a less frequent detection is required. If the camera motion can be somehow measured, an appropriate threshold can be set to declare an incoming frame as a reference frame. An intuitive way to quantify the camera motion is to calculate the magnitude of translation.

The authors in [14], [25] have shown that, given a homography matrix, it can be decomposed to estimate the translation. The decomposition yields four solutions in general out of which only two are physically possible. However, each of these solutions has the same magnitude of translation. We make use of this information to detect the reference frame. The solutions for decom-

[4] For fitting homography to keypoints using RANSAC, we used the code available at http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/Robust/ransacfithomography.m

position[5] of a homography H are obtained using the method in [25].

Let t be the translation vector of one of the obtained solutions such that $t = [t_1, t_2, t_3]^T$. Then the magnitude of translation is given by $|t| = \sqrt{t_1^2 + t_2^2 + t_3^2}$. Also, let δ_r be the threshold for translation. Considering the first video frame as a reference *ref*, a homography along with the translation between the reference and every incoming frame f_i is calculated. If the corresponding translation is greater than δ_r, then the frame f_i is declared as a reference. For the new incoming frames, f_i becomes the reference frame. This method is given in algorithm 3. The translation threshold δ_r is set experimentally[6] and depends on the frame size.

Algorithm 3 Detection of reference frame

```
% Let the $i^{th}$ video frame be denoted by $f_i$, such that
the video consists of total $k$ frames. If $R_i := 1$ then $f_i$ is a
reference frame.

% Initialization
$R_1 = 1; R_i := 0\ \forall i := 2, \ldots, k$.
$ref := f_1$. {reference frame.}

% Update $R_i$
for $i := 2$ to $k$ do
    $suc := f_i$. {subsequent frame.}
    Estimate translation $t$ between ref & suc.
    if $|t| \geq \delta_r$ then
        $R_i := 1$.
        $ref := suc$.
    end if
end for

% Result
return $R_i\ \forall i := 1, \ldots, k$.
```

4.3 Tracking and inpainting cracked regions across frames

Every incoming frame is tested for being a reference frame. When a reference frame is encountered, the cracked regions that appear in this frame are detected using the proposed method described in section 3. If the incoming frame is not a reference frame, then a homography with respect to the previous frame is estimated using the procedure described in section 4.1. The estimated homography is used to track the cracked regions across these frames. For a reference frame, pixels in the newly detected cracked regions are independently inpainted using the technique proposed in [10], while for a non-reference frame, the inpainted regions from the previous frame are copied to the tracked regions. The tracking of cracked regions is described below.

For a pair of temporally adjacent frame f_{i-1} and f_i, the crack pixels at locations (x_i, y_i) in f_i can be tracked using the corresponding locations of crack pixels (x_{i-1}, y_{i-1}) in f_{i-1} as given below in equation (5).

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = H_i \begin{bmatrix} x_{i-1} \\ y_{i-1} \\ 1 \end{bmatrix}, \tag{5}$$

where (x'_i, y'_i, z'_i) are the homogeneous coordinates for the point (x_i, y_i) in frame f_i, such that $x_i = \frac{x'_i}{z'_i}$ and $y_i = \frac{y'_i}{z'_i}$, and H_i denotes the homography between frames f_{i-1} and f_i.

Here, it may happen that estimated coordinates x_i and y_i are real numbers. These are rounded to the nearest integers so that we have the tracked pixels at integer locations. For simplicity, let the integer-rounded location coordinates be denoted by (x_i, y_i). Setting these damaged locations to 1 with all other locations set to a value of 0, a crack-mask consisting of 1's and 0's is constructed for the frame f_i. Since homography introduces geometric distortions, it may happen that some narrow cracked regions detected in the frame f_{i-1} may become disjoint regions in the newly constructed crack-mask, which leads to some part of the cracked regions being missed out. In order to avoid this, we use morphological closing on the crack-mask to connect the nearby disjoint regions. The crack-mask now gives the locations of the tracked cracks in the frame f_i that correspond to the crack regions detected in the frame f_{i-1}. Figure 12 illustrates the tracking of cracked regions.

We now describe how an incoming frame is processed. The first video frame f_1 being a reference frame is independently inpainted after identifying the cracked regions in it. Any subsequent incoming frame f_i may or may not be a reference frame depending on the camera motion. For both cases, we use the above procedure along with equation (5) to track cracked regions from f_{i-1} to f_i. Let P_i denote the binary image consisting of the cracked regions tracked from frame f_{i-1} to frame f_i.

In case f_i is not a reference frame, it can be inpainted by filling up the location of the tracked crack pixels (i.e. $\{(x_i, y_i) | P_i(x_i, y_i) = 1\}$). This is achieved by simply copying the values of the corresponding pixels from the inpainted version of the previous frame f_{i-1}. Here, the frames are temporally adjacent and the

[5] For decomposition of estimated homography, we have used the implementation available at http://cs.gmu.edu/∼kosecka/examples-code/homography2Motion.m

[6] The details of selecting threshold δ_r are given in section 6.

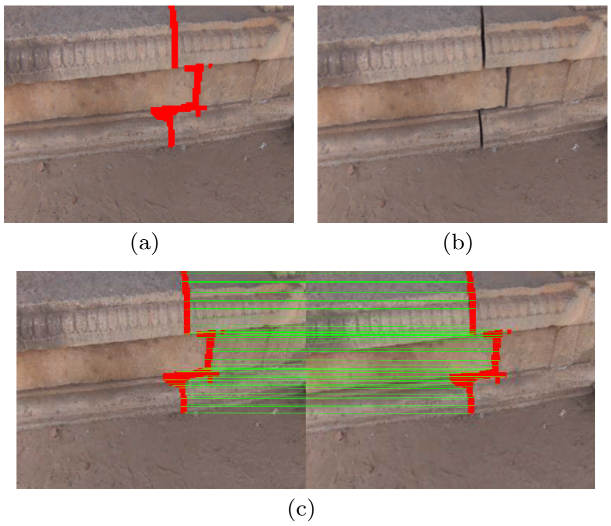


Fig. 12 Tracking detected regions using the estimated homography matrix. (a) Detected damaged regions in the frame f_{i-1}; (b) frame f_i; (c) tracked cracks in the frame f_i. Green lines show the mapping of few points on the boundary of crack regions, while the detected and tracked cracked regions using SIFT features are shown in red.

change in intensity of corresponding pixels is negligible. Also note that the selected translation threshold δ_r is small enough so that the change in intensity of corresponding pixels across frames within this translation is also negligible. Thus, the copying of pixel values across subsequent frames does not introduce any seam.

Since the homography matrix H_i is non-singular, its inverse H_i^{-1} exists. Therefore, the crack pixels at locations (x_i, y_i) and the corresponding locations (x_{i-1}, y_{i-1}) from the previous frame f_{i-1}, must be related as follows,

$$\begin{bmatrix} x'_{i-1} \\ y'_{i-1} \\ z'_{i-1} \end{bmatrix} = H_i^{-1} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{6}$$

where $(x'_{i-1}, y'_{i-1}, z'_{i-1})$ are the homogeneous coordinates for the point (x_{i-1}, y_{i-1}), such that $x_{i-1} = \frac{x'_{i-1}}{z'_{i-1}}$ and $y_{i-1} = \frac{y'_{i-1}}{z'_{i-1}}$. Since x_i and y_i were rounded to integers, we may obtain the corresponding x_{i-1} and y_{i-1} as real numbers. The intensity at this location is obtained by considering the first-order integer location neighbourhood and using the bilinear interpolation. It may be noted that inpainting performed in this manner across is almost insensitive to small changes in the morphologically closed crack-mask due to directly copying the values from the previously inpainted regions.

If the incoming frame f_i is a reference frame, then crack detection is performed independently. However, since only the newly appearing cracked pixels need to be inpainted, we first calculate the binary image P_i consisting of the cracked regions tracked from the previous frame f_{i-1}. Now, let B_i denote the crack detected binary image corresponding to f_i obtained using the method described in section 3. Then, the binary image Q_i consisting only the newly appearing cracked pixels is given by,

$$Q_i(x_i, y_i) = \begin{cases} 1, \ B_i(x_i, y_i) - P_i(x_i, y_i) > 0, \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

Now, an initial inpainting of the reference frame f_i is achieved by using the inpainted version of the previous frame f_{i-1} and the binary image P_i. The locations (x_{i-1}, y_{i-1}) in frame f_{i-1} corresponding to the pixels at locations $\{(x_i, y_i)|P_i(x_i, y_i) = 1\}$ are obtained using the relation in equation (6). Similar to inpainting a non-reference frame as described above, the pixels at locations (x_i, y_i) are filled up by copying values from the corresponding pixels at locations (x_{i-1}, y_{i-1}) to obtain the initial inpainted image. The newly detected cracked pixels given by the binary image Q_i are the holes to be filled up in the initially inpainted image. The final inpainted version of the reference frame is obtained by

Algorithm 4 Video frame inpainting

```
% Let the $i^{th}$ video frame be denoted by $f_i$, such that the video consists of total $k$ frames. $R_i := 1$ denotes $f_i$ is a reference frame. Let $A_i$ denote the inpainted version of frame $f_i$.

% Initialization
Detect damaged regions in $f_1$ to get $B_1$.
Set threshold $\delta_0 := |B_1|$.
Perform inpainting on $f_1$ using $B_1$ to get $A_1$.

% Loop
for $i := 2$ to $k$ do
  Extract SIFT descriptors and homography $H_i$.
  Calculate $P_i$ by tracking damaged regions.
  if $R_i := 1$ then
    Detect damaged regions in $f_i$ to get $B_i$.
    Calculate $Q_i$ using $P_i$ and $B_i$.
    if $|Q_i| :\leq \delta_0$ then
      Calculate $S_i$ using $P_i$ and $B_i$.
      Fill pixels $\{(x_i, y_i)|S_i(x_i, y_i) = 1\}$ using $A_{i-1}$ to get initial inpainted image init.
      Perform inpainting on init using $Q_i$ to get $A_i$.
    else
      $R_i := 0$, $R_{i+1} := 1$.
      Fill tracked pixels $\{(x_i, y_i)|P_i(x_i, y_i) = 1\}$ using $A_{i-1}$ to get $A_i$.
    end if
  else
    Fill tracked pixels $\{(x_i, y_i)|P_i(x_i, y_i) = 1\}$ using $A_{i-1}$ to get $A_i$.
  end if
end for

% Result
return Inpainted frames $A_i$ $\forall i := 1, \ldots, k$.
```

using the method proposed in [10] considering the initial inpainted image and the binary image Q_i as inputs. An example for performing inpainting when a new reference frame appears is shown in figure 21 along with the experimental results in section 6.

It may happen that a detected reference frame is highly blurred or noisy due to an unstable camera motion. In such a case, the crack detection method described in section 3 may fail and detect many regions as cracked. This can be avoided by simply thresholding the number of pixels in the newly detected cracked regions. Assuming that the number of pixels in the cracked regions do not vary substantially across the reference frames or whenever a new reference frame is encountered, we set a threshold δ_0 based on the number of cracked pixels detected in the first frame. This is a valid assumption because, while the camera moves and new cracked regions enter a frame, some pixels of the previously detected cracked regions may exit. Also, even if the cracked pixels do not exit, we expect only few new cracked pixels to enter. Let $|Q_i|$ denote the number of newly detected cracked pixels in the frame f_i and $|B_1|$ denote the number of cracked pixels detected in the first frame. Then, for a reference frame f_i, if we have $|Q_i| > \delta_0$ (such that $\delta_0 = 0.5 * |B_1|$), the frame f_i is treated as a non-reference frame and inpainting is performed accordingly. Also, the frame f_{i+1} is set as a reference frame, provided f_i is not the last frame. The complete procedure for inpainting video frame is given in algorithm 4.

5 Measuring temporal consistency of the inpainted video

The quality of a processed image/video is usually quantified in terms of some metric by comparing the image/video with an undistorted source. For example, in video compression, the quality of a video reconstructed at a receiver is measured by comparing it with the original video transmitted by the sender. However, in some applications the original source or reference is not available for comparison. Video inpainting is one such application in which missing regions in frames need to be filled up and hence a reference for comparison is not available. In such a case, the objective quantification of the video quality is based on no-reference video quality assessment (NR VQA) metrics viz. blockiness, bluriness and sudden local changes [13], [36], [37].

Blockiness gives the measure of spurious blocking artefacts usually present at the boundary of coding blocks. Higher the value, higher is the strength of blocking artefacts. Blurriness gives a measure of blur in the frame. It is estimated based on the average width of strong edges in the frame. More the blur, higher is the average width and higher is the blurriness value. The blockiness and blurriness are measured individually for each video frame. For a video sequence, the blockiness and blurriness values are taken as the average over all the frames. The calculation of blockiness and blurriness metrics has been proposed in [13].

The sudden local changes across the video frames can be measured using the technique given in [37], [36]. Here, the average value of the discrete cosine transform (DCT) coefficients of every coding block in the difference frame is calculated. Mean of the highest 10% average DCT coefficients is considered as the measure of sudden local change between two frames. For a video sequence, these values are averaged over all the pairs of adjacent frames.

The techniques described above estimate the video quality directly from the processed video, without considering the unprocessed video. However, in an application like video inpainting, some information from the unprocessed video also can be used to quantify the quality of the processed video. The temporal consistency measure between two videos that we introduce here indicates similarity of between two videos in terms of the optical flow. Intuitively, to obtain a temporally plausible inpainted video, the optical flow of the input video should be maintained on inpainting, provided the objects to be inpainted are stationary. In other words, the optical flow between every pair of temporally adjacent frame in input and corresponding pair of frames in the inpainted video should be similar. The inpainting of only the stationary object is a valid assumption for inpainting videos of heritage monuments. With this cue, the optical flow between every pair of adjacent frames in both input as well as inpainted video can be estimated and used to quantify the quality of the inpainted video. An example of temporal consistency in terms of optical flow is shown in figure 13. The optical flow can be estimated by using the classic method proposed by Lucas and Kanade [24].

Let $L_0(i)$ and $D_0(i)$ be the magnitude and direction, respectively, of the optical flow between the i^{th} and $i+1^{th}$ frames in the input video. Similarly, let $L_1(i)$ and $D_1(i)$ be the magnitude and direction, respectively, of the optical flow between the i^{th} and $i+1^{th}$ frames in the inpainted video. Both L and D are vectorized using lexicographical ordering. Then, the temporal consistency between i^{th} and $i+1^{th}$ frames is given by the Pearson's correlation coefficient $r(i)$ as follows [21].

$$r(i) = \frac{1}{l-1} \sum_{j=1}^{l} \frac{(K_0^j(i) - \bar{K}_0)(K_1^j(i) - \bar{K}_1)}{\sigma_0(i)\sigma_1(i)}, \tag{8}$$

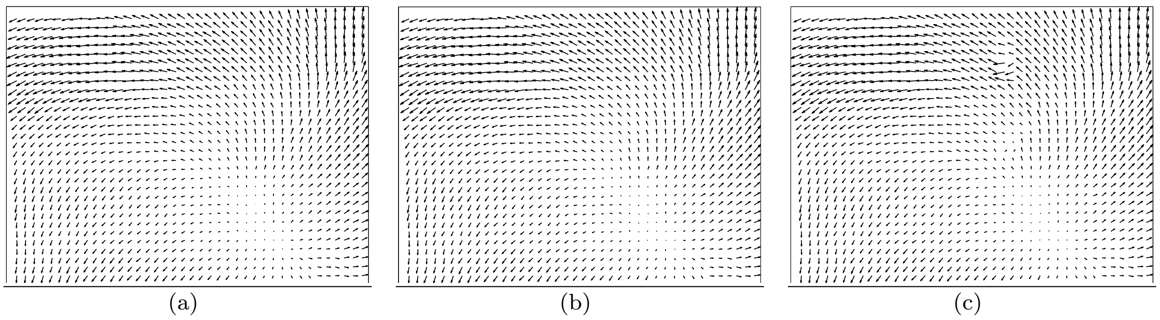


Fig. 13 Optical flow between a pair of temporally adjacent frames in (a) input video, (b) auto-inpainted video using proposed method, (c) video generated by auto-inpainting every frame independently. The optical flow in (a) and (b) appear to be similar while some haphazard orientations in the optical flow are observed in (c).

where K can be the vector of magnitude (L) or direction (D), $\bar{K}$ and σ are mean and standard deviation of K respectively, and l represents the length of K. The value $r(i) = +1$ indicates perfect positive correlation, $r(i) = -1$ indicates perfect negative correlation while $r(i) = 0$ for un-correlated data. The average value of r for all the pairs of adjacent frames then gives the temporal consistency between the input and the inpainted videos. A higher average value of r indicates higher temporal consistency.

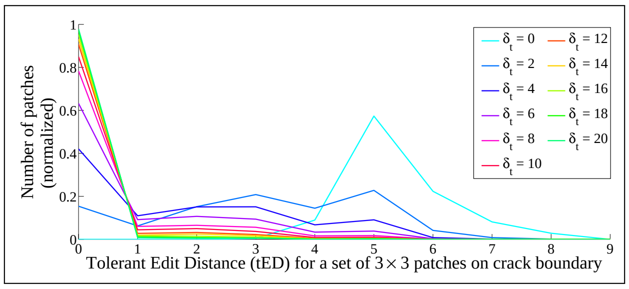


Fig. 14 Curves for varying tolerance values δ_t.

6 Experimental Results

In this section, we present the results of our proposed technique for automatic detection and inpainting of crack-like damaged regions, on images and videos captured by us, as well as on images downloaded from various sources on the Internet. These images and videos contain cracked regions in the form of breaks splitting the objects/scene. The inpainted results show the effectiveness of our proposed methods.

For all our experiments to detect cracked region, we have considered patches Φ_p of size 3×3. Patches of larger sizes did not significantly improve the detection. In calculation of the tolerant edit distance, we have set the tolerance value $\delta_t = 10$ based on the following experimentation. We considered many patches at the boundary of known cracked regions from a number of images, along with their corresponding non-overlapping adjacent patches. For each of these patches, we calculated the tolerant edit distances by varying values of δ_t. Curves of tolerant edit distance versus normalized number of patches, corresponding to every δ_t were plotted, as shown in figure 14. Since the patches belonged to crack boundaries, we have higher edit distance (i.e. $\delta_t = 0$). Increasing the value of δ_t reduces the sensitivity and, therefore, only large variations can be detected. It is observed that for $\delta_t = 10$, sufficiently large variations were detected and further increasing δ_t did not change the curve significantly.

The size of structuring element for morphological closing used for filling in large gaps depends on the image size. For an image of size $M \times N$, the size of structuring element is taken to be $(\max(M, N)/360 + \min(M, N)/270)$. In the proposed method for auto-inpainting in videos, the frames are of size 270×360. However, the method also works on videos with larger frames at an expense of increased computations.

Detection of the reference frames is based on the translation threshold δ_r. For frames of size 270×360, we conducted the following experiment. We manually selected two frames viz. 1) the frame in which a cracked region has completely appeared and 2) the frame in which the next cracked region begins. For every such pair of frames, translation was calculated. Conducting the experiment on a number of videos revealed that the average value of $\delta_r = 25$ can be used to detect new incoming cracked regions. However, the problem with this threshold is that, while a part of the newly appearing cracked region gets detected successfully, the remaining part which appears in subsequent frames is never detected. For successful detection of the complete cracked regions, a lower value of threshold is required. By decreasing the threshold from 25 to 0, we found $\delta_r = 5$ to be an appropriate threshold for successful detection of the complete cracked regions. Also note that the intensity change in corresponding pixels across the frames within this small translation is negligible. This enables a seamless copying of pixel values when propagating the already inpainted cracked regions across subsequent frames. We have, therefore, set $\delta_r = 5$.

We present the results of our experiments on heritage site images in figures 15–20. The input images shown in figures 15(a)–19(a) are of size 684×912, while the image in figure 20(a) is of size 400×300. We also show a comparison of the results obtained by our crack detection method with those obtained using the techniques proposed by Amano [1], Turakhia et al. [43] and Padalkar et al. [30]. It may be noted that the results for the technique in [1] are the best possible, obtained after fine-tuning the parameters.

The images considered for our experiments contain cracked objects/scenes for which no ground truth is available that would show how these objects/scenes existed in their entirety. We, therefore, rely on the observations made by volunteers and consider the regions selected by them as cracked regions that need to be inpainted. In order to determine the suitability of the resulting detection for the use by inpainting algorithms, the popularly known recall and precision metrics are considered. These are defined as follows [48].

$$\begin{aligned} Recall &= \frac{|Ref \bigcap Dect|}{|Ref|} \quad , \\ Precision &= \frac{|Ref \bigcap Dect|}{|Dect|}. \end{aligned} \tag{9}$$

Here, Ref are the pixels declared to be in the cracked regions by volunteers and $Dect$ are the pixels detected by the algorithm to be in the cracked regions. However, for providing an insight into the robustness of our proposed algorithm, we use a slightly different precision measure defined as, $Precision = \frac{|Ref_{conn}|}{|Dect|}$. Here, Ref_{conn} are those pixels detected in $Dect$ that are connected to cracked regions in Ref. Higher value of *Precision* indicates that a large number of detected pixels indeed belong to the cracked regions, while a higher value of *Recall* indicates that a large number of cracked pixels have been detected. For a mask to be suitable for use to an inpainting algorithm, it is, therefore, desired to have the *Recall* value nearer to 1. On the other hand, a low value for *Precision* indicates that pixels more than the desired ones have been detected, which increases the area to be inpainted. If a large number of pixels other than the desired target pixels get detected, then many regions in the image get inpainted unnecessarily, which may lead to undesired results. Nevertheless, if a mask with low *Recall* value is used for inpainting, information from the undetected target regions may propagate inside the detected regions, leading to poor inpainting results.

The performance of the proposed method in comparison with the techniques suggested by Amano [1], Turakhia et al. [43] and Padalkar et al. [30] in terms of *Recall*, *Precision* and execution time, for input images in figures $15 - 20$ is given in table 1. The results in figures $15 - 20$ show that the cracked detection results obtained using our proposed technique are similar to the detection performed manually by volunteers. This is also evident from the performance table 1 where we observe that both *Recall* and *Precision* values for the detected cracked regions in most of these images are nearer to 1, indicating that the desired crack pixels have been detected with high accuracy.

On the other hand, the technique proposed in [1] results in detection of either (a) pixels that do not correspond to the desired cracked regions or (b) too many pixels around the desired cracked regions. The later leads to unnecessary inpainting of many regions, modifying the large undamaged regions in the image, which is not desirable. Moreover, it slows down the inpainting as the inpainting process is computationally expensive. The cracked regions are clearly visible in the inpainted results shown in figures 15(g), 17(g) and 18(g), while the results shown in figures 16(g), 19(g) and 20(g) display poor inpainting. Although the technique in [1] is good for detecting any alteration to the photograph (like overlay text), our proposed method is fast and more suitable when it comes to detection of cracked in the photographed scene/object. The results of our crack detection method are at par with and in some cases better than those obtained using the techniques in [30] and [43]. Yet, our method is significantly faster, more accurate and the inpainted results are convincing.

In videos, an example of inpainting whenever a new reference frame is encountered is shown in figure 21. We present the results of the proposed automatic crack inpainting method on videos captured by us from the heritage site at Hampi, Karnataka, India. These results are shown in figures 22–25. Although the videos were captured at only one heritage site, the proposed method is generic and should work for other heritage site videos. As an example, we show one more result on a video of the McConkie Ranch Petroglyphs near Vernal, Utah, USA, in figure 26 that demonstrates the effectiveness of the proposed method. This video was uploaded by an enthusiast on the popular streaming site YouTube [38].

From the reported results, we can observe that by using the proposed method the detected cracked regions are effectively tracked and plausibly inpainted to

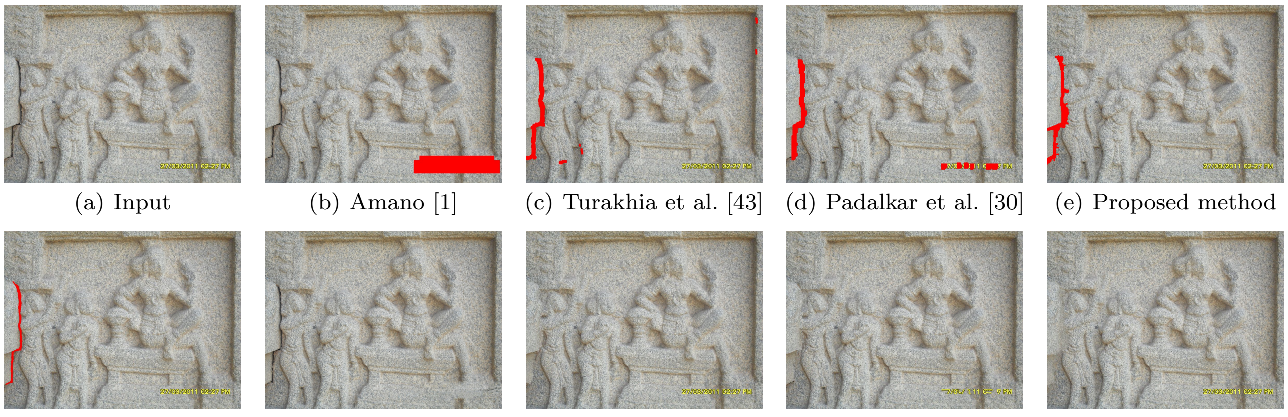

(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method
(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 15 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.

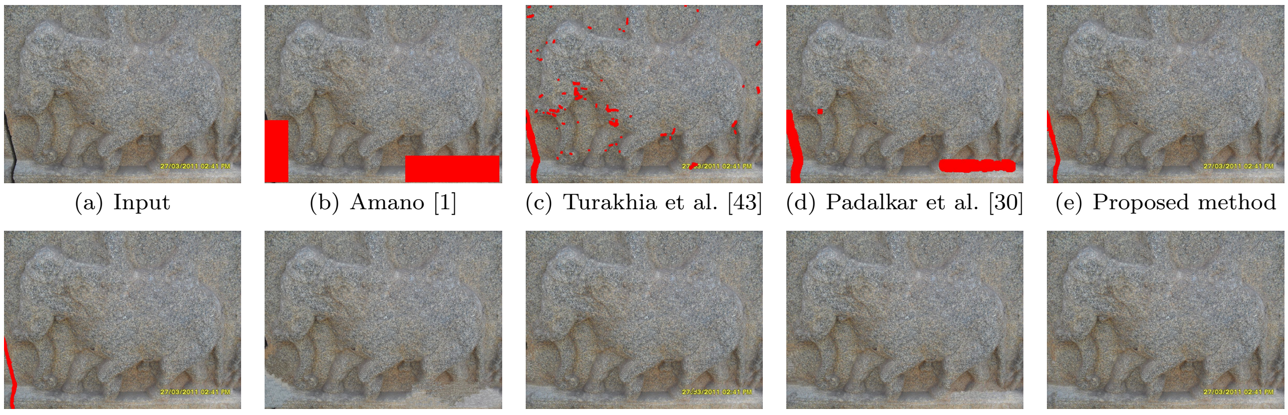

(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method
(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 16 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.

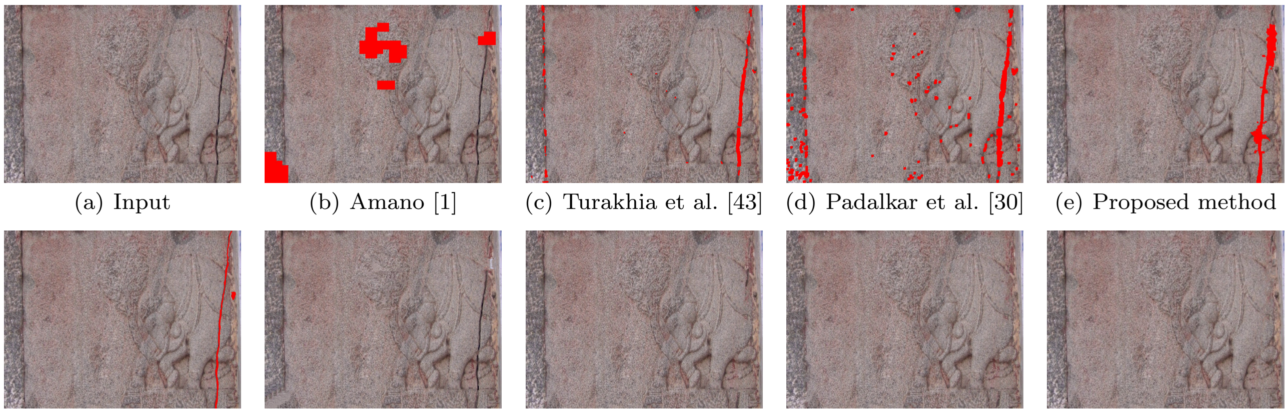
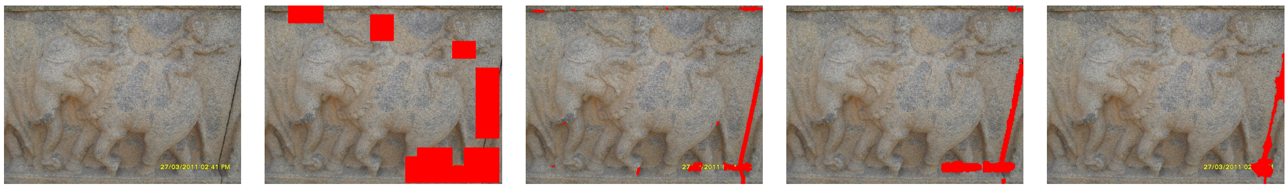

(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method
(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 17 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.


(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method

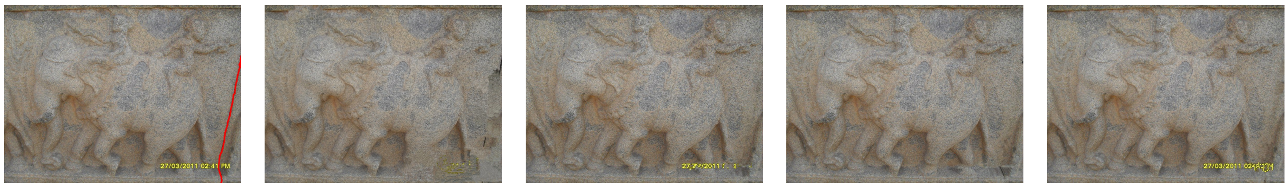
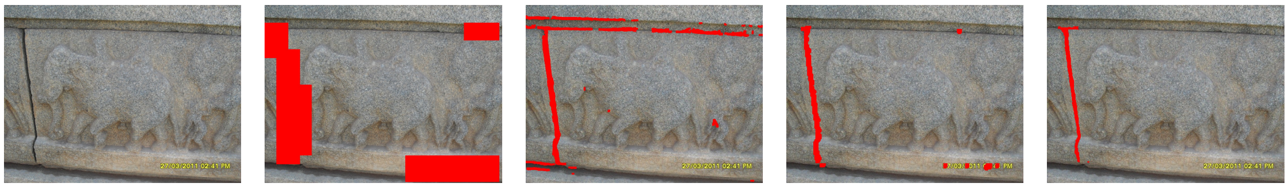

(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 18 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.


(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method

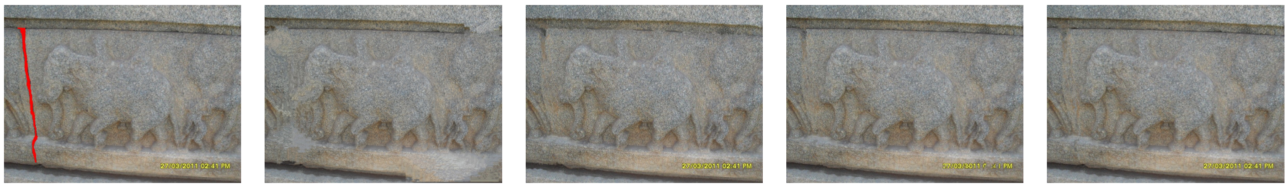
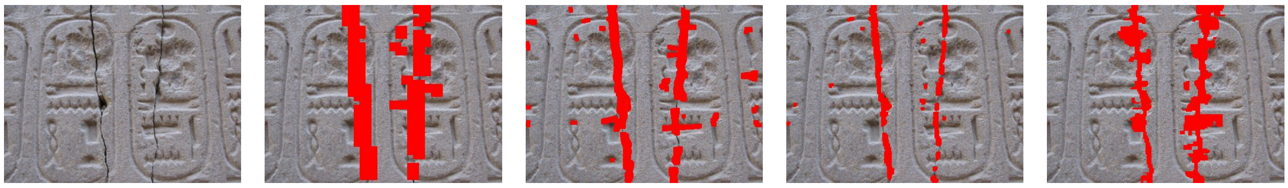

(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 19 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.


(a) Input (b) Amano [1] (c) Turakhia et al. [43] (d) Padalkar et al. [30] (e) Proposed method

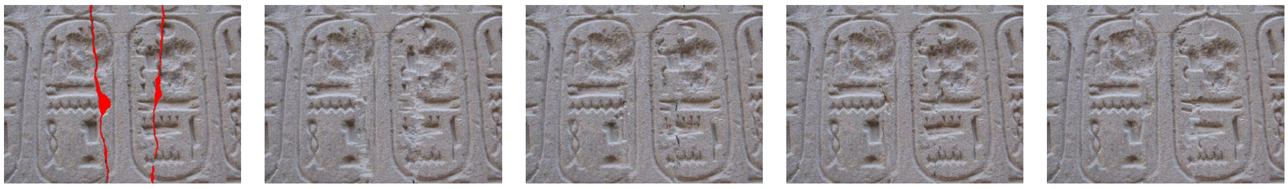

(f) Selection by volunteers (g) Inpainting of (b) (h) Inpainting of (c) (i) Inpainting of (d) (j) Inpainting of (e)

Fig. 20 Detection and inpainting of cracked regions in images. The detected regions in (b)–(e) and the manual selection by volunteers in (f) are shown in red color. Results of inpainting the regions detected in (b)–(e) are shown in (g)–(j), respectively.

Table 1 Performance comparison in terms of Recall, Precision and execution time.

Input	# Crack pixels	Amano [1]			Turakhia et al. [43]			Padalkar et al. [30]			Proposed technique		
		Recall	Precision	Time (sec)	Recall	Precision	Time (sec)	Recall	Precision	Time (sec)	Recall	Precision	Time (sec)
Fig. 15(a)	3494	0.000	0.000	109.0	0.988	0.887	21.65	0.953	0.743	04.51	0.990	1.000	03.62
Fig. 16(a)	3819	0.918	0.370	13.02	0.970	0.390	22.22	1.000	0.422	14.54	0.969	1.000	03.32
Fig. 17(a)	5162	0.046	0.068	302.3	0.749	0.678	23.29	0.863	0.392	12.77	0.840	0.997	05.02
Fig. 18(a)	2997	0.783	0.737	12.06	0.999	0.728	25.16	0.921	0.678	04.80	0.990	0.997	03.49
Fig. 19(a)	5435	1.000	0.579	19.01	0.974	0.974	29.92	0.987	0.857	04.89	0.985	0.996	04.77
Fig. 20(a)	2276	0.966	0.949	1500	0.932	0.949	13.64	0.808	0.898	05.23	0.952	0.989	07.44

Table 2 Comparison of proposed method with frame-by-frame auto-inpainting, in terms of blockiness (**A**), bluriness (**B**), sudden local change (**C**) and temporal consistency in optical flow's direction (**D**) and magnitude (**E**).

Video	Proposed method					Frame-by-frame auto-inpainting				
	A	B	C	D	E	A	B	C	D	E
Video1 (Fig. 22)	0.1125	5.1020	1.0737	0.9529	0.7501	0.1296	5.1073	1.3126	0.5064	0.2496
Video2 (Fig. 23)	0.1034	4.1261	1.5459	0.6671	0.9604	0.1270	4.2057	1.9463	0.1978	0.4148
Video3 (Fig. 24)	0.2975	4.3382	1.2908	0.9979	0.5424	0.2292	4.3666	1.5322	0.1862	0.6134
Video4 (Fig. 25)	0.1453	4.6306	1.8454	0.8173	0.9678	0.1473	4.7223	2.0858	0.2009	0.8946
Video5 (Fig. 26)	0.1582	3.1264	2.0559	0.5821	0.9654	0.1662	3.1586	2.7768	0.2301	0.9381

get a seamless video. Although there exist approaches for semi-automatic inpainting of unwanted elements in videos [46] and video inpainting under constrained camera motion [32], it may be noted that, to the best of our knowledge, there does not exist any approach that demonstrates automatic video inpainting under unconstrained camera motion with no moving objects. Our approach handles these cases and we, therefore, do not show any comparison with the approaches in [46],[32]. However, we do compare the proposed approach with auto-inpainting done in a frame-by-frame fashion. The results of the proposed method, along with auto-inpainting performed individually on every frame are shown in figures 22–25.

An objective comparison of the proposed method with frame-by-frame auto-inpainting is presented in table 2. A video with higher blockiness and blur has higher value of the blockiness and blurriness metrics [13], respectively. For a temporally plausible video, the sudden local change [37], [36] is less while the temporal consistency measure has a higher value. From table 2 we observe that the proposed method performs better in terms of blockiness, sudden local change and temporal consistency, which is in accordance with the results in figures 22–25.

The implementation details along with the timing information are presented as follows: for images, the calculation of tolerant Edit Distance which involves comparison of many patches is implemented in C (Matlab MEX) while the rest of the method is implemented in Matlab. For a 684×912 sized image, the initial detection takes about 1.5 seconds on a Windows 7 Professional operating system with 32 bit Intel Core i5, 2.5GHz CPU and 3 GB RAM. The remaining detection time is spent on refinement, which again is a C (Matlab MEX) implementation. However, in the same setup, the process for inpainting (for example the regions detected in figure 15(e)) requires about 37 seconds, which is also a C (Matlab MEX) implementation. Therefore, at present the implementation does not execute in real-time and needs to be performed offline. In future, if a faster inpainting method is developed, the implementation could run in nearly real-time.

In the case of videos also the implementation is done in Matlab. Here, the cracked region detection in reference frames and independent inpainting of newly detected cracked pixels are achieved with the C (Matlab MEX) implementation used above for images. For frames of size 270×360 (for example the video corresponding to figure 22), the inpainting of reference frames takes nearly 1.5–2 seconds. This includes the time required for tracking and inpainting from previously detected cracked regions (about 0.6 seconds) followed by initial detection, refinement and inpainting of the newly detected cracked pixels. The first frame, however, required about 4.5 seconds for initial detection, refinement and inpainting. Note that the size of the frame here is 270×360. Subsequent (non-reference) frames take about 0.08 seconds to complete tracking and inpainting from previously detected cracked regions, which is very fast when compared to independent inpainting of each frame. In this case, a considerable amount of time is required for the inpainting operation in reference frames. In real-time, this introduces a lag in the video.

Although major computational steps are implemented in C (Matlab MEX), our implementation is not an optimized version but a proof of concept of the proposed method. Having said that, we are optimistic about an implementation for mobile phones in order to use the method directly at heritage sites. This is because of the quick inpainting of subsequent (non-reference) frames. A real-time on-the-fly inpainting of the video frames could be possible with an implementation optimized for the hardware of mobile phones.

7 Conclusion

In this paper we have presented a technique that can automatically detect cracked regions and use these regions for inpainting. By comparing non-overlapping patches using the tolerant edit distance measure introduced here, our method initially localizes the cracked regions. Further, using an active contour-based segmentation, the results are refined to accurately detect the cracked regions. Based upon this crack detection method, we build up a method that can be used to automatically detect and inpaint cracked regions in videos captured at heritage sites. The new cracked pixels detected in the reference frames are inpainted independently. The homography estimated between two temporally adjacent frames is used to track and the cracked regions in subsequent frames. The reported results suggest that the method can be used to auto-inpaint the cracked regions captured in heritage site videos. In future, we aim to extend this detection method to perform simultaneous on-the-fly detection and inpainting, which can be used to build an immersive walk-through system.

References

1. Amano, T.: Correlation based image defect detection. In: Proceedings of the 18th International Conference on Pattern Recognition, *ICPR '06*, vol. 01, pp. 163–166. IEEE Computer Society, Washington, DC, USA (2006)
2. Ammouche, A., Riss, J., Breysse, D., Marchand, J.: Image analysis for the automated study of microcracks in concrete. Cement and Concrete Composites **23**(23), 267 – 278 (2001). Special Theme Issue on Image Analysis
3. Bendels, G.H., Guthe, M., Klein, R.: Free-form modelling for surface inpainting. In: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa, AFRIGRAPH '06, pp. 49–58. ACM, New York, NY, USA (2006)
4. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH '00, pp. 417–424. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
5. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. International Journal of Computer Vision **74**(1), 59–73 (2007)
6. Chan, T., Vese, L.: Active contours without edges. Image Processing, IEEE Transactions on **10**(2), 266–277 (2001)
7. Chang, R.C., Sie, Y.L., Chou, S.M., Shih, T.K.: Photo defect detection for image inpainting. In: Proceedings of the Seventh IEEE International Symposium on Multimedia, ISM '05, pp. 403–407. IEEE Computer Society, Washington, DC, USA (2005)
8. Cornelis, B., Rui, T., Gezels, E., Dooms, A., Piurica, A., Platia, L., Cornelis, J., Martens, M., Mey, M.D., Daubechies, I.: Crack detection and inpainting for virtual restoration of paintings: The case of the ghent altarpiece. Signal Processing **93**(3), 605 – 619 (2013). Image Processing for Digital Art Work
9. Criminisi, A., Pérez, P., Toyama, K.: Object removal by exemplar-based inpainting. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on **2**, 721 (2003)
10. Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on Image Processing **13**, 1200–1212 (2004)
11. El-Hakim, S.F.: Semi-automatic 3d reconstruction of occluded and unmarked surfaces from widely separated views. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Greece, pp. 143–148 (2002)
12. El-Hakim, S.F., MacDonald, G., Lapointe, J.F., Gonzo, L., Jemtrud, M.: On the digital reconstruction and interactive presentation of heritage sites through time. In: VAST 2006: The 7th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage, Nicosia, Cyprus, 2006. Proceedings, pp. 243–250. Eurographics Association (2006)
13. Farias, M., Mitra, S.: No-reference video quality metric based on artifact measurements. In: Image Processing, 2005. ICIP 2005. IEEE International Conference on, vol. 3, pp. III–141–4 (2005)
14. Faugeras, O., Lustman, F.: Motion and structure from motion in a piecewise planar environment. Tech. Rep. RR-0856, INRIA (1988)
15. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
16. Grossauer, H.: A combined pde and texture synthesis approach to inpainting. European Conference on Computer Vision **4**, 214–224 (2004)
17. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2 edn. Cambridge University Press, New York, NY, USA (2003)
18. Jeannin, S., Divakaran, A.: MPEG-7 visual motion descriptors. IEEE Transactions on Circuits and Systems for Video Technology **11**(6), 720–724 (2001). DOI 10.1109/76.927428
19. de Kadt, C., Gain, J., Marais, P.: Revisiting district six: a case study of digital heritage reconstruction from archival photographs. In: Proceedings of the 6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, AFRIGRAPH '09, pp. 13–21. ACM, New York, NY, USA (2009)
20. Kang, H.W., Shin, S.Y.: Creating walk-through images from a video sequence of a dynamic scene. Presence:

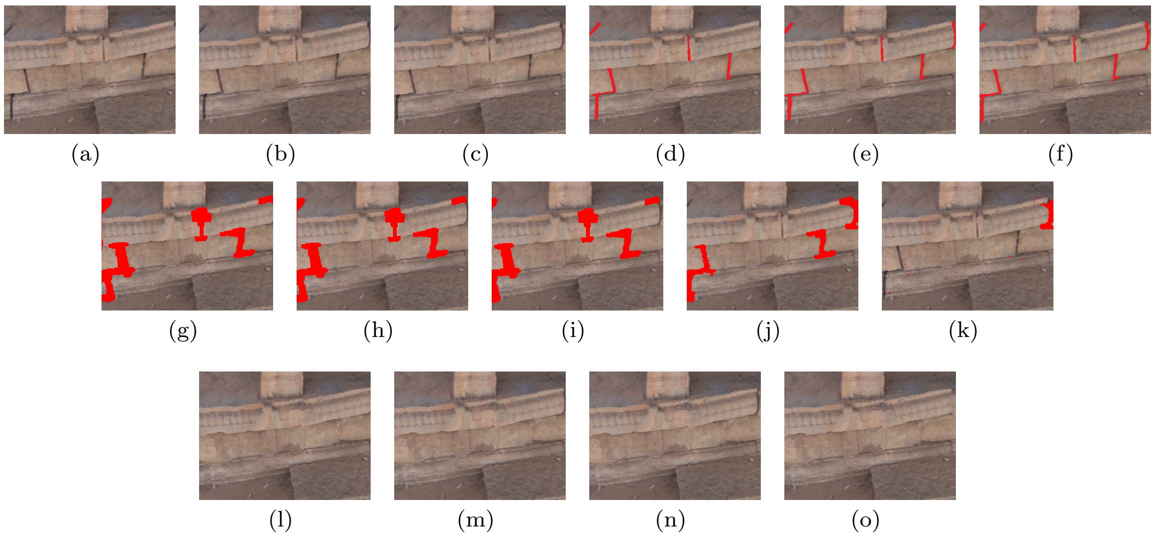


Fig. 21 Inpainting a newly appearing reference frame f_i. (a),(b),(c) show frames f_{i-2}, f_{i-1} and f_i, respectively, while the cracked regions selected by volunteers corresponding to these frames are shown in (d),(e),(f); the cracked regions corresponding to (a),(b),(c) tracked from detected cracks in previous frames are shown in (g),(h),(i); independent crack detection in f_i is shown in (j), while the newly appearing cracked pixels in (j) with respect to (i) are displayed in (k); the inpainted versions of f_{i-2}, f_{i-1}, f_i obtained by copying pixels from respective previous inpainted frames are shown in (l),(m),(n); final inpainted version of f_i obtained after inpainting the newly detected pixels given in (k) is shown in (o). Note that the crack visible near the right side in (n) is filled in (o) by independently inpainting the pixels shown in (k).

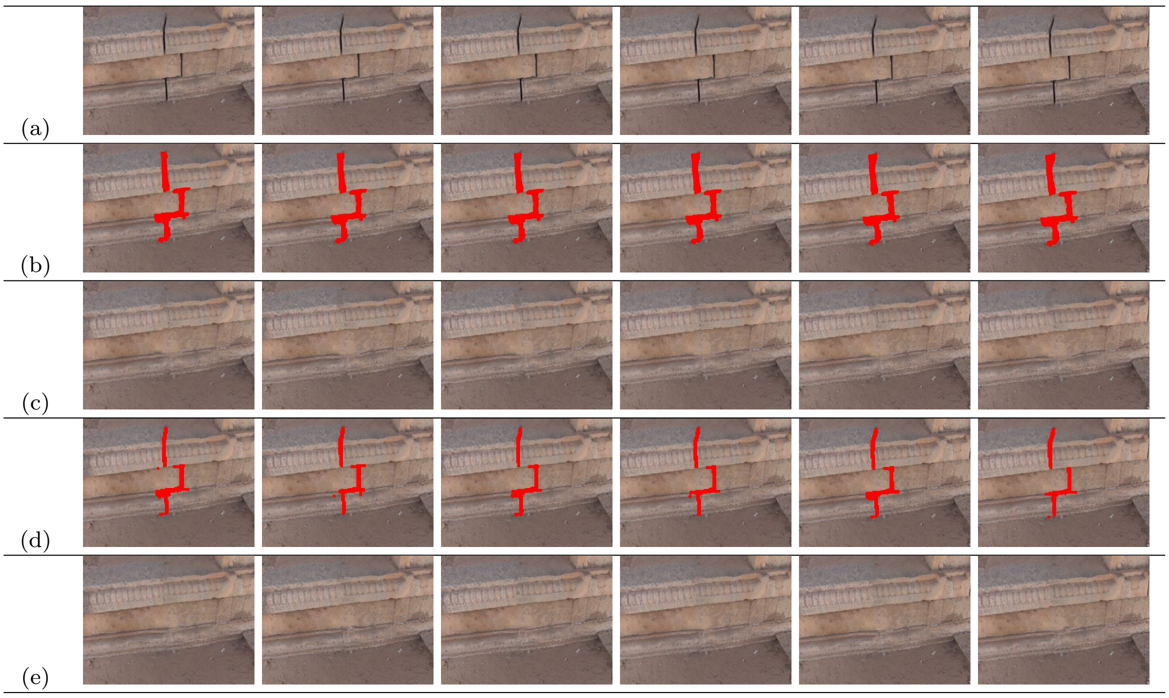


Fig. 22 Result of auto-inpainting cracked regions in video frames. (a) Input frame sequence, left most frame is the reference frame; (b) cracked regions auto-detected in the reference frame tracked using SIFT and homography; (c) inpainted frames corresponding to frames in (b); (d) cracked regions auto-detected independently in each frame; (e) inpainted frames corresponding to frames in (d).

Telepresence, and Automotive Environments **13**, 638–655 (2004)

21. Kenney, J.F.: Mathematicals of Statistics. Van Nostrand (1954)

22. Kovesi, P.D.: MATLAB and Octave functions for computer vision and image processing. Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia (2005). Available from:

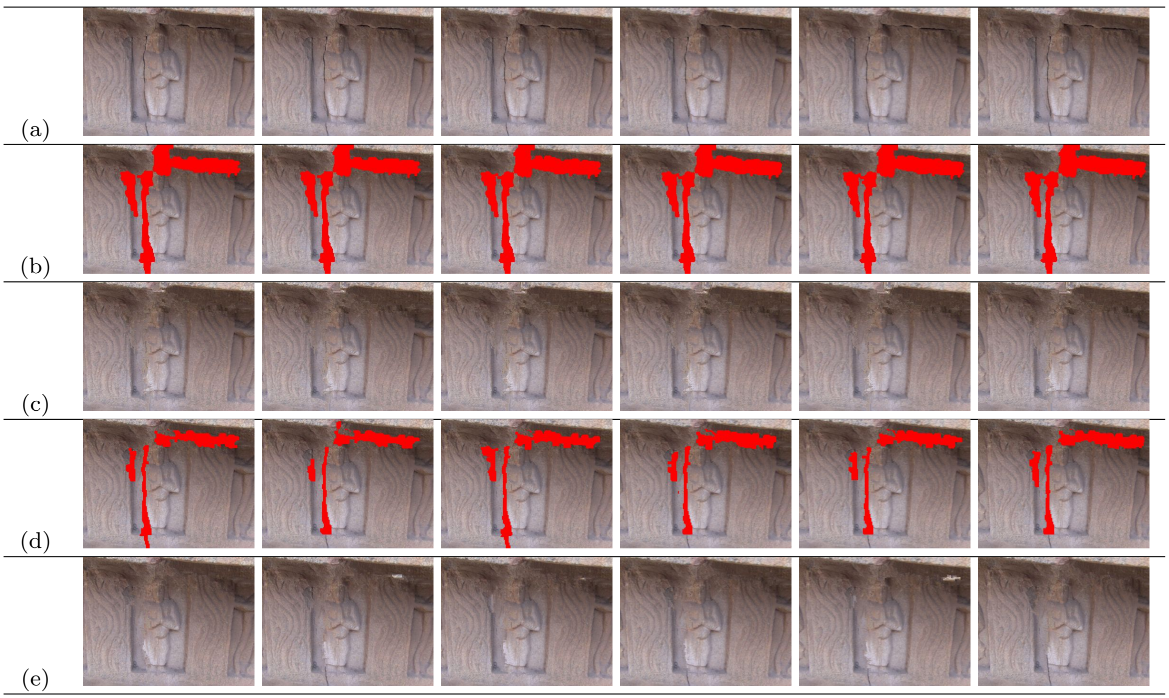


Fig. 23 Result of auto-inpainting cracked regions in video frames. (a) Input frame sequence, left most frame is the reference frame; (b) cracked regions auto-detected in the reference frame tracked using SIFT and homography; (c) inpainted frames corresponding to frames in (b); (d) cracked regions auto-detected independently in each frame; (e) inpainted frames corresponding to frames in (d).

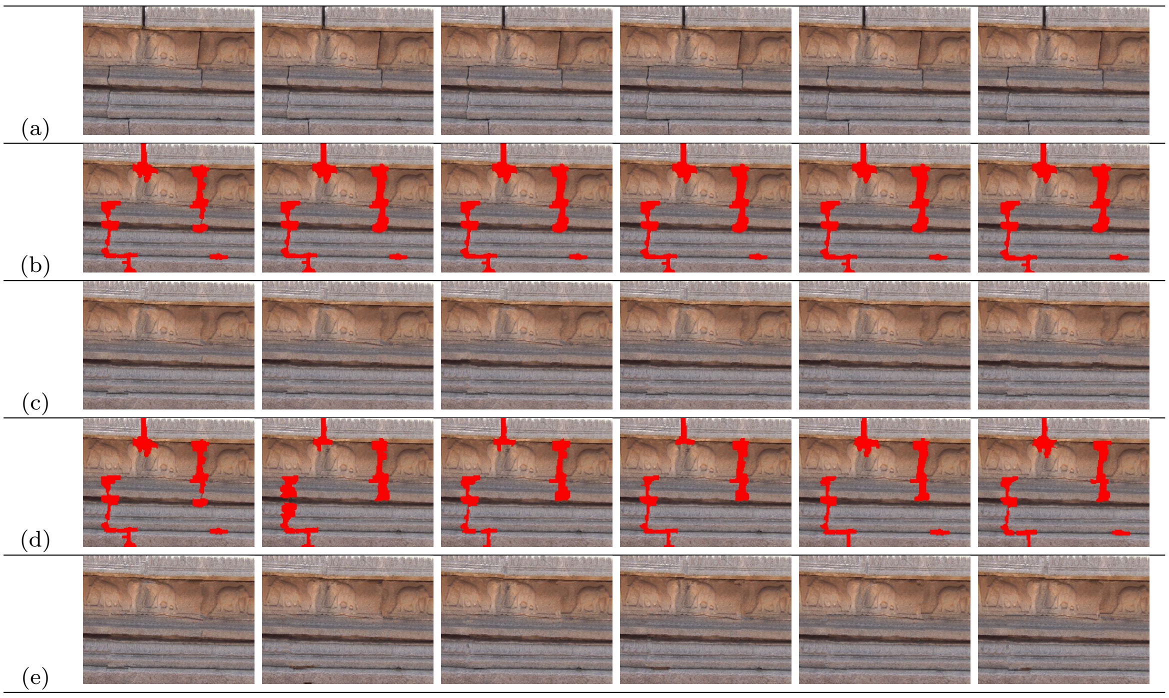


Fig. 24 Result of auto-inpainting cracked regions in video frames. (a) Input frame sequence, left most frame is the reference frame; (b) cracked regions auto-detected in the reference frame tracked using SIFT and homography; (c) inpainted frames corresponding to frames in (b); (d) cracked regions auto-detected independently in each frame; (e) inpainted frames corresponding to frames in (d).

<http://www.csse.uwa.edu.au/∼pk/research/matlabfns/robust/ransacfithomography.m>

23. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer

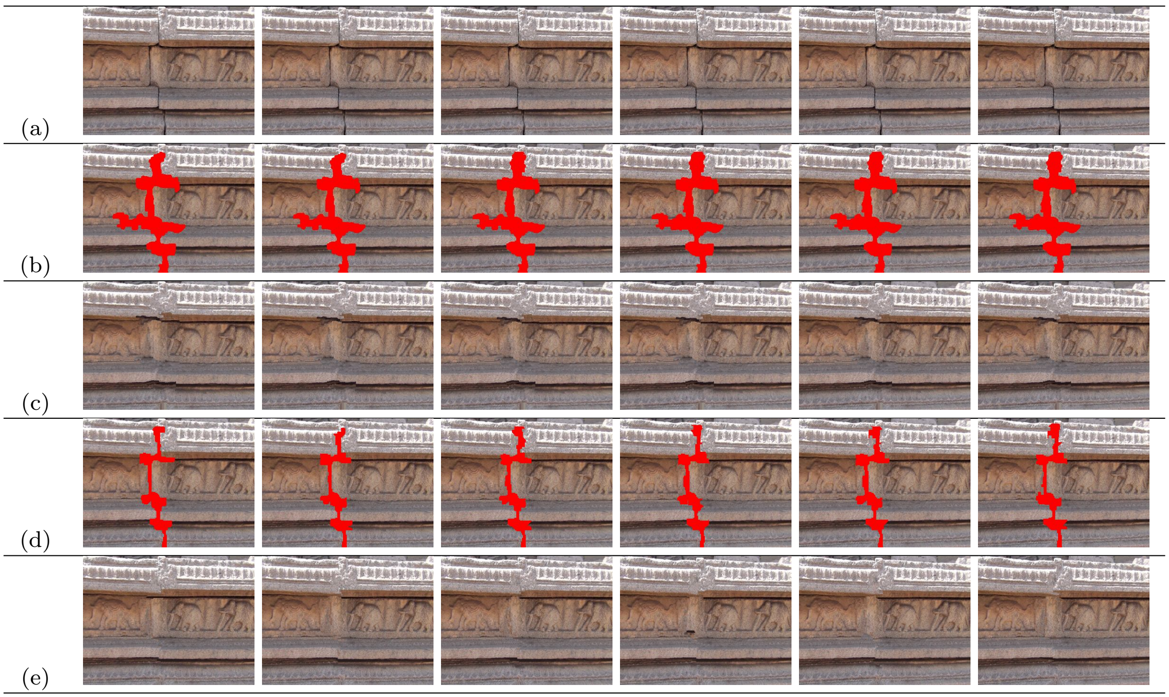


Fig. 25 Result of auto-inpainting cracked regions in video frames. (a) Input frame sequence, left most frame is the reference frame; (b) cracked regions auto-detected in the reference frame tracked using SIFT and homography; (c) inpainted frames corresponding to frames in (b); (d) cracked regions auto-detected independently in each frame; (e) inpainted frames corresponding to frames in (d).

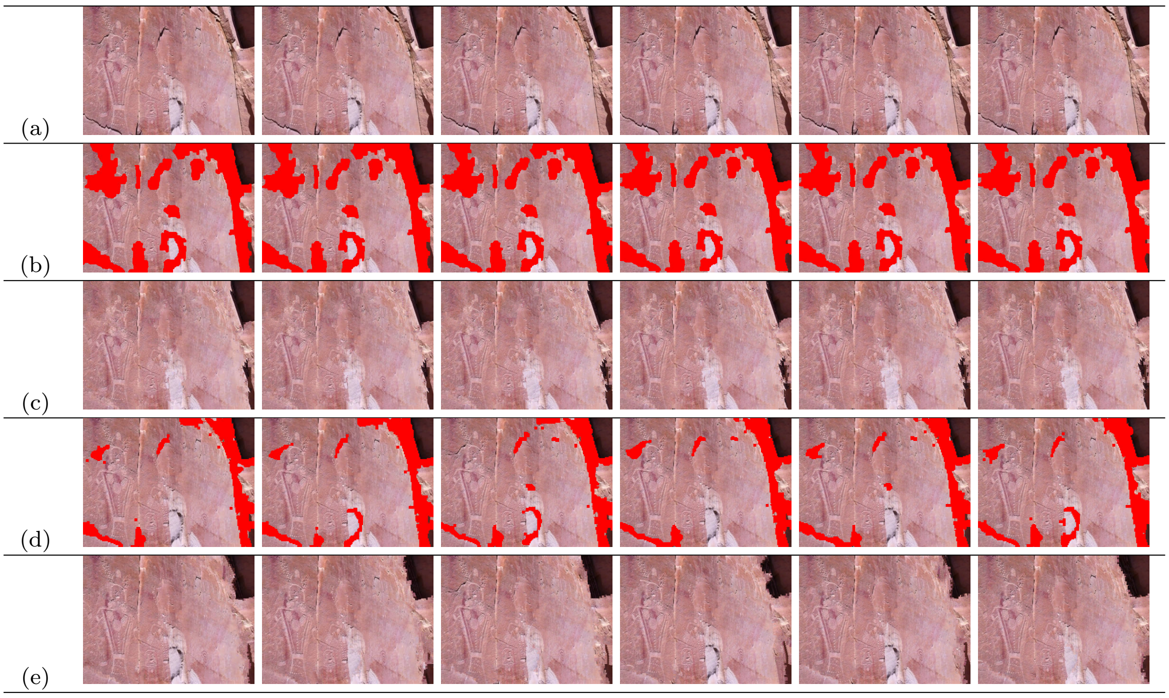


Fig. 26 Result of auto-inpainting cracked regions in video frames. (a) Input frame sequence, left most frame is the reference frame; (b) cracked regions auto-detected in the reference frame tracked using SIFT and homography; (c) inpainted frames corresponding to frames in (b); (d) cracked regions auto-detected independently in each frame; (e) inpainted frames corresponding to frames in (d).

Vision **60**(2), 91–110 (2004)

24. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Pro-

ceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81, pp. 674–679 (1981)
25. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S.: An Invitation to 3-D Vision: From Images to Geometric Models. SpringerVerlag (2003)
26. Masnou, S., Morel, J.M.: Level lines based disocclusion. Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on pp. 259–263 (1998)
27. McCloskey, S., Langer, M., Siddiqi, K.: Removal of partial occlusion from single images. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(3), 647 – 654 (2011). DOI 10.1109/TPAMI.2010.187
28. Oliveira, M.M., Bowen, B., Mckenna, R., sung Chang, Y.: Fast digital image inpainting. In: Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001, pp. 261–266. ACTA Press (2001)
29. Padalkar, M.G., Vora, M.V., Joshi, M.V., Zaveri, M.A., Raval, M.S.: Identifying vandalized regions in facial images of statues for inpainting. In: New Trends in Image Analysis and Processing–ICIAP 2013, *Lecture Notes in Computer Science*, vol. 8158, pp. 208–217. Springer Berlin Heidelberg (2013)
30. Padalkar, M.G., Zaveri, M.A., Joshi, M.V.: Svd based automatic detection of target regions for image inpainting. In: J.I. Park, J. Kim (eds.) Computer Vision - ACCV 2012 Workshops, *Lecture Notes in Computer Science*, vol. 7729, pp. 61–71. Springer Berlin Heidelberg (2013)
31. Parmar, C.M., Joshi, M.V., Raval, M.S., Zaveri, M.A.: Automatic image inpainting for the facial images of monuments. In: Proceedings of Electrical Engineering Centenary Conference 2011, pp. 415–420. IISc Bangalore, India (2011)
32. Patwardhan, K.A., Sapiro, G., Bertalmío, M.: Video inpainting under constrained camera motion. IEEE Transactions on Image Processing **16**(2), 545–553 (2007)
33. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Transactions on Graphics **22**(3), 313–318 (2003)
34. Ringot, E., Bascoul, A.: About the analysis of microcracking in concrete. Cement and Concrete Composites **23**(23), 261 – 266 (2001). Special Theme Issue on Image Analysis
35. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": Interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**(3), 309–314 (2004)
36. Saad, M., Bovik, A., Charrier, C.: Blind prediction of natural video quality. Image Processing, IEEE Transactions on **23**(3), 1352–1365 (2014). DOI 10.1109/TIP.2014.2299154
37. Saad, M.A., Bovik, A.C.: Blind quality assessment of videos using a model of natural scene statistics and motion coherency. In: Asilomar Conference on Signals, Systems, and Computers, pp. 332–336 (2012)
38. Scholes, S.: Mcconkie ranch petroglyphs near vernal, utah (2011). URL https://www.youtube.com/watch?v=jmewuqEXTK8. [Accessed 01 Sept. 2014]
39. Shibata, T., Iketani, A., Senda, S.: Image inpainting based on probabilistic structure estimation. In: Proceedings of the 10th Asian conference on Computer vision - Volume Part III, ACCV'10, pp. 109–120. Springer-Verlag, Berlin, Heidelberg (2011)
40. Shih, T.K., Tang, N.C., Yeh, W.S., Chen, T.J., Lee, W.: Video inpainting and implant via diversified temporal continuations. In: Proceedings of the 14th annual ACM international conference on Multimedia, MULTIMEDIA '06, pp. 133–136. ACM, New York, NY, USA (2006)
41. Sikora, T.: MPEG digital video-coding standards. Signal Processing Magazine, IEEE **14**(5), 82–100 (1997)
42. Tamaki, T., Suzuki, H., Yamamoto, M.: String-like occluding region extraction for background restoration. International Conference on Pattern Recognition **3**, 615–618 (2006)
43. Turakhia, N., Shah, R., Joshi, M.: Automatic crack detection in heritage site images for image inpainting. In: Eighth Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP), p. 68 (2012)
44. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974)
45. Wang, Z., Zhou, F., Qi, F.: Inpainting thick image regions using isophote propagation. In: Image Processing, 2006 IEEE International Conference on, pp. 689–692 (2006). DOI 10.1109/ICIP.2006.312428
46. Wexler, Y., Shechtman, E., Irani, M.: Space-time completion of video. Pattern Analysis and Machine Intelligence, IEEE Transactions on **29**(3), 463–476 (2007)
47. Wu, J., Ruan, Q.: Object removal by cross isophotes exemplar-based inpainting. In: Proceedings of the 18th International Conference on Pattern Recognition, ICPR '06, pp. 810–813. IEEE Computer Society, Washington, DC, USA (2006)
48. Zou, Q., Cao, Y., Li, Q., Mao, Q., Wang, S.: Cracktree: Automatic crack detection from pavement images. Pattern Recognition Letters **33**(3), 227 – 238 (2012)