Real-Time Video Texture Synthesis for Multi-Frame Capsule Endoscopy Visualization

Ady Ecker

adyecker @ cs.toronto.edu

Abstract

We present a real-world application of real-time video texture synthesis for capsule endoscopy.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Overview

Over the last two decades, texture synthesis methods [WLK*09] produced extremely realistic results. Yet, their real-world use is often limited to offline, human-supervised applications, and their complexity limits their use in video displays. In this paper, we present a real-world application of texture synthesis for reviewing capsule endoscopy videos. Wireless capsule endoscopy is a non-invasive way for visualizing the digestive tract. The widespread use of capsule endoscopy is limited by the relatively long time it takes gastroenterologists to review these videos. A common way to reduce the review time is to display multiple video frames simultaneously, side by side. However, cameras with wide-angle lens, such as fisheye or endoscopic cameras, produce rounded images. Hence, only a limited amount of warping to close the holes between these images can be applied without introducing large distorsions. For smooth viewing, texture synthesis is needed to fill-in the holes between the rounded frames (a real application can also display the frames without the synthesis for refined medical diagnosis).

Naive attempts to fill holes between rounded frames, e.g. interpolating the colors in the fill area, catch the eye's attention. Standard inpainting approaches could be acceptable to fill a static gap, but dynamic motions outside the fill area cause the fill area to pop out. Patch-based texture synthesis approaches have several limitations. Graph-cuts [WLK*09] are undesirable since we do not want to cut out any imaged area and reduce the tissue coverage. Non-parametric sampling methods, e.g. [WSI07], are too expensive.

Our method processes a single frame at a time. Suppose we warped a patch close to the fill area onto the fill area (see Fig. 1, right). By plugging-in a patch from a fixed location, we automatically gain several advantages. First, the placed patch is temporally coherent with the previous and next frames. Second, the temporal motion direction in the filled area is likely to be similar to its neighborhood. Third, this operation is very fast. However, plugging a neighboring patch may create a noticeable edge at its boundary. Just smoothing this edge will again attract the eye since the texture and motion patterns will be different at this edge compared to the rest of the image. The problem of hiding this edge, while preserving the texture, is closely related to image blending. However, we blend images without overlap area, and apply processing to one side only (the filled part).

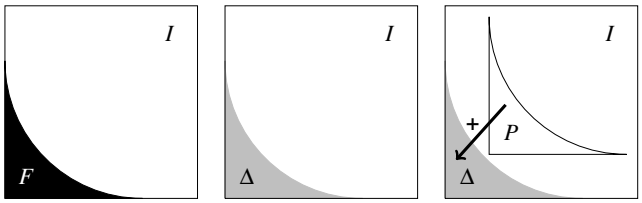

Figure 1: *Texture synthesis of a corner. Left: After warping the image I, we want to fill the empty region F. Middle: First we compute the Δ-map. Right: the fill-in is $F = P + \Delta$, where P is a fixed close image patch. The Δ-map is constructed in-place before adding P.*

In [Pel81,PGB03], it was suggested to add a smooth offset map Δ to the filled region that will eliminate the edge while preserving the texture, as shown in Fig. 1. The Δ-map is a solution of a Poisson system with boundary conditions specified by the edge. The solution is computed by a relaxation process [Pel81] or a Poisson solver [PGB03], both are computationally expensive. In [FHL*09] it was realized that solving the Poisson equation is unnecessary. We just need to find some smooth Δ-map, e.g. by mean value coordinates. However, in mean value coordinates interpolation, the value of a point in the interior region depends on every point on the boundary polygon. This approach is still expensive (required

Algorithm 1: Edge-hiding between image and placed patch

Input : Warped image I, a patch P to place in the fill area F
Output: Δ-map and filled region $F = P + \Delta$ (as in Fig. 1)

$\Delta(i,j) \longleftarrow \emptyset$
Compute the distance transform D for each pixel in F from the boundary of I
foreach *pixel* $(i,j) \in F$ *in increasing order of* $D(i,j)$ **do**
  if $D(i,j) < 2$ **then**
    $m(i,j) = mean\big(I(i_1,j_1)$ *such that* $(i_1,j_1) \in I, |i-i_1| \le 1, |j-j_1| \le 1\big)$
    $\Delta(i,j) = m(i,j) - P(i,j)$
  else
    $m(i,j) = mean\big(\Delta(i_1,j_1)$ *such that* $|i-i_1| \le 1, |j-j_1| \le 1, \Delta(i,j) \ne \emptyset\big)$
    $\Delta(i,j) = m(i,j) \cdot d$, *where d is a decay factor*
  end
end
return $F = P + \Delta$

a GPU), and is more natural for closed polygonal regions than blending images with an open polygonal boundary.

Our main insight is that the Δ-map computation process can be simplified and accelerated if we perform a propagation-and-averaging process *from the boundary inwards*. This is similar to inpainting methods [Tel04], but *instead of filling-in by smoothing the image, we smooth the Δ-map that is added to an image patch*. Pseudo-code of the algorithm is given in Alg. 1. Its operation is illustrated in Fig. 2. The algorithm is efficient because for each pixel we only average about four values. This is similar to a single iteration of Peleg's algorithm [Pel81], except that we propagate the information efficiently inwards, and we use a decay factor to have the Δ-map approach zero away from the boundary. Our algorithm can be parallelized naturally by processing concurrently one set of pixels at increasing distances from the boundary at a time, but this was unnecessary for our application. An example of the warped and filled layout generated by our method is shown in Fig. 3 (obviously, the smoothness of the video cannot be appreciated). The synthesis is at a quality level that viewers that were not told in advance are unlikely to notice that parts in the video were synthesized.

2. Conclusions

While non-rectangular images are not uncommon in medical imaging, the problem of displaying effectively multiple non-rectangular video frames received little attention. Since warping circular images into shapes with straight corners introduces a noticeable distortion, we limited the amount of warping and proposed a *new, simple, fast, and practical* method to fill-in gaps in videos. The insight that made the synthesis practical for real-time video is that instead of smoothing the image and losing the texture, we should smooth a Δ-map that is added to an image patch. This can be implemented efficiently by inward propagation rather than solving the expensive Poisson equations. Our technique can be applied to other applications such as interactive image cloning [FHL*09] and image and video blending.

$I_{0,6}$	$I_{1,6}$	$I_{2,6}$	$I_{3,6}$	$I_{4,6}$	$I_{5,6}$	$I_{6,6}$
$I_{0,5}$	$I_{1,5}$	$I_{2,5}$	$I_{3,5}$	$I_{4,5}$	$I_{5,5}$	$I_{6,5}$
$I_{0,4}$	$I_{1,4}$	$I_{2,4}$	$I_{3,4}$	$I_{4,4}$	$I_{5,4}$	$I_{6,4}$
$\Delta_{0,3}$	$I_{1,3}$	$I_{2,3}$	$I_{3,3}$	$I_{4,3}$	$I_{5,3}$	$I_{6,3}$
$\Delta_{0,2}$	$\Delta_{1,2}$	$I_{2,2}$	$I_{3,2}$	$I_{4,2}$	$I_{5,2}$	$I_{6,2}$
$\Delta_{0,1}$	$\Delta_{1,1}$	$\Delta_{2,1}$	$I_{3,1}$	$I_{4,1}$	$I_{5,1}$	$I_{6,1}$
$\Delta_{0,0}$	$\Delta_{1,0}$	$\Delta_{2,0}$	$\Delta_{3,0}$	$I_{4,0}$	$I_{5,0}$	$I_{6,0}$

Figure 2: *Computation of the Δ-map (before adding the image patch). In the first stage, we set Δ for the boundary pixels (grayed) so that the addition of the image patch will not create a strong edge. For example, $\Delta_{1,1} = I_{2,2} - I_{4,4}$, $\Delta_{2,1} = \frac{1}{3}(I_{2,2} + I_{3,2} + I_{3,1}) - I_{5,4}$. In the second stage, we propagate the Δ values inwards, e.g. $\Delta_{0,1} = \frac{1}{3}(\Delta_{0,2} + \Delta_{1,2} + \Delta_{1,1}) \cdot d$, $\Delta_{1,0} = \frac{1}{4}(\Delta_{0,1} + \Delta_{1,1} + \Delta_{2,1} + \Delta_{2,0}) \cdot d$. The values of Δ are written in place and used immediately. The algorithm is applied to each RGB color channel independently.*

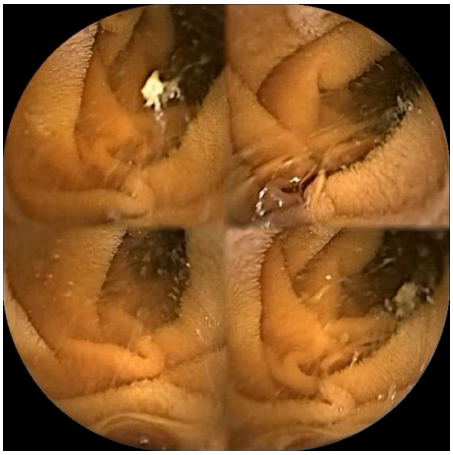

Figure 3: *Four frames layout. Each of the four frames is filled at three corners (in the figure's center and middle of the sides).*

References

[FHL*09] FARBMAN Z., HOFFER G., LIPMAN Y., COHEN-OR D., LISCHINSKI D.: Coordinates for instant image cloning. In *ACM Transactions on Graphics (TOG)* (2009), vol. 28, ACM, p. 67. 1, 2

[Pel81] PELEG S.: Elimination of seams from photomosaics. *Computer Graphics and Image Processing 16*, 1 (1981), 90–94. 1, 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 313–318. 1

[Tel04] TELEA A.: An image inpainting technique based on the fast marching method. *Journal of graphics tools 9*, 1 (2004), 23–34. 2

[WLK*09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G., ET AL.: State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (2009), pp. 93–117. 1

[WSI07] WEXLER Y., SHECHTMAN E., IRANI M.: Space-time completion of video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 29*, 3 (2007), 463–476. 1